Sistema embebido para la gestión energética basado en tecnología IoT y servicios cloud

Grado en Ingeniería Robótica

Trabajo Fin de Grado

Autor:
Borja Sánchez Valdés
Tutor/es:
Francisco Javier Ferrandez Pastor

Universitat d'Alacant
Universidad de Alicante

Julio 2019

Sistema embebido para la gestión energética basado en tecnología IoT y servicios cloud

Autor
Borja Sánchez Valdés

Tutor/es
Francisco Javier Ferrandez Pastor
Tecnología informática y computación

Grado en Ingeniería Robótica

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Julio 2019

Preámbulo

"La motivación de este Trabajo de fin de grado viene por la necesidad de mejorar la eficiencia energética, aplicando capacidades adquiridas en mi grado, y para adelantar la llegada del futuro con las Smart Homes."

Agradecimientos

Este trabajo no habría sido posible sin el apoyo y el estímulo de Dani, trabajador en Elecgy Solutions, el cual siempre estuvo dispuesto a ayudarme en los momentos de desesperación, y gracias al cual comencé a realizar este TFG.

También me gustaría agradecer a Francisco Javier, mi tutor, y a la Escuela Politécnica Superior de la Universidad de Alicante, sin sus ayudas y consejos no habría sido lo mismo.

No puedo terminar sin agradecer a mi familia, en cuyo estímulo constante y amor he confiado a lo largo de mis años en la Universidad. Estoy agradecido también a de mis amigos. Su constante apoyo ha sido vital a lo largo de todos estos años.

Es a ellos a quien dedico este trabajo.

A mi familia y amigos, recordad que yo no valgo nada sin vosotros.

Si consigo ver más lejos
es porque he conseguido auparme
a hombros de gigantes

Isaac Newton.

Índice general

Índice de figuras

Índice de Códigos

1 Introducción.

1.1 Contextualización.

El contexto del proyecto se enmarca en la integración de tecnologías embebidas hardware, protocolos de comunicación IoT[1] (Internet of Things), servicios cloud y la aplicación de paradigmas de Inteligencia Artificial en escenarios definidos por instalaciones de consumo y generación eléctrica en edificios y grupos de viviendas (micro smartgrids). El contexto global se define en la figura 1.1.

1.2 Antecedentes.

En consecuencia de la Industria 4.0 y el impulso de las tecnologías del Internet de las cosas (IoT), se han dado una gran cantidad de avances en el ámbito de la energía, mejorando sobre todo la eficiencia de la misma y reduciendo las pérdidas. Pero antes de nada debemos mencionar qué implica el IoT en la eficiencia energética. [1]

- "Cosas": Transformadores eléctricos, motores eléctricos, compresores de aire, lámparas, hornos, secadores, quemadores, cámaras frigoríficas, bombas de calor de oficinas, calderas, en general, todo equipo que consume energía.
- Sensores: analizadores de redes eléctricas, contadores de gas, de agua, sensores de humedad, temperatura, presión, luminosidad, viscosidad, opacidad, composición química, todo componente o equipo capaz de medir una variable energética o una variable que puede afectar al consumo energético.
- Actuadores: relés, motores, en general aquellos equipos que permiten modificar el proceso o los equipos del proceso con el objeto de mejorar su eficiencia.
- Controladores: software/hardware de control para la actuación sobre un sistema o equipo con objeto de mejorar su eficiencia energética.

Todos estos sistemas estarán conectados entre sí y con internet, mejorando el rendimiento, tal y como se ve en la figura 1.1.

Estas mejorías han llevado a la creación de lo que se conoce como una Red Eléctrica Inteligente o REI, que ha acabado derivando en la creación de Casas Inteligentes o Smart Homes. Este trabajo de fin de grado (TFG), titulado "Sistema embebido para la gestión energética basado en tecnología IoT y servicios cloud", empleará el dispositivo embebido Raspberry Pi 3 para la adquisición de datos, subida de los mismos a la nube, toma de decisiones en la nube, y manejo de actuadores en función de las decisiones tomadas. También

[1] IoT: Internet of Things, emplearemos estas siglas para referirnos al internet de las cosas.

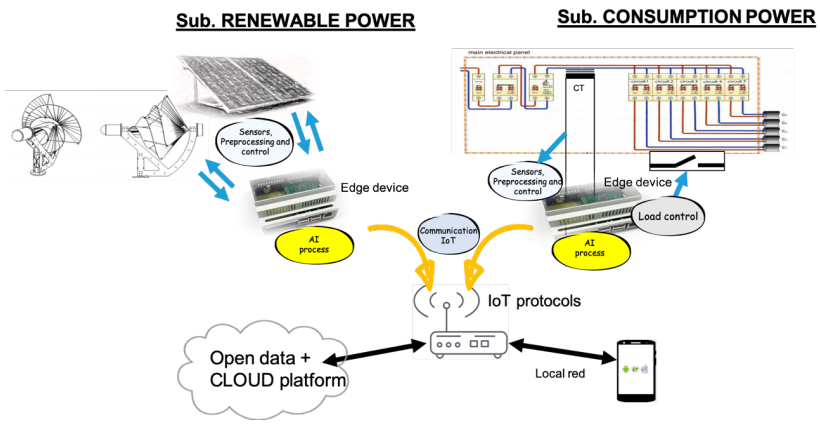


Figura 1.1: Contexto global. Fuente: cedida por el grupo de innovación UCIE Ars Innovatio de la Universidad de Alicante, a través del tutor de este proyecto

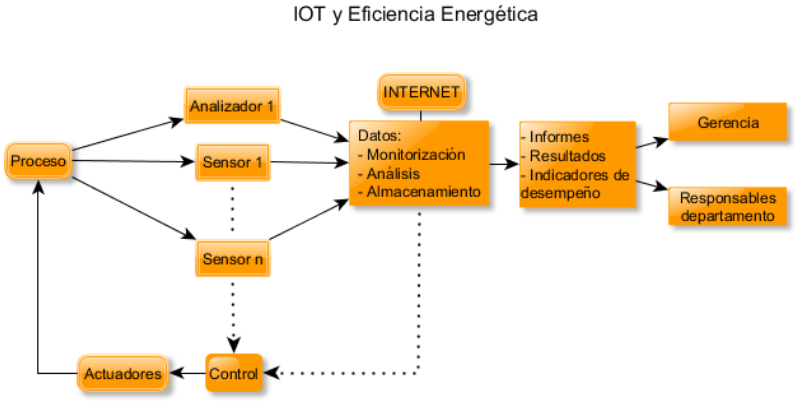


Figura 1.2: Sencillo esquema de cómo afectaría el IoT a la eficiencia energética. [1]

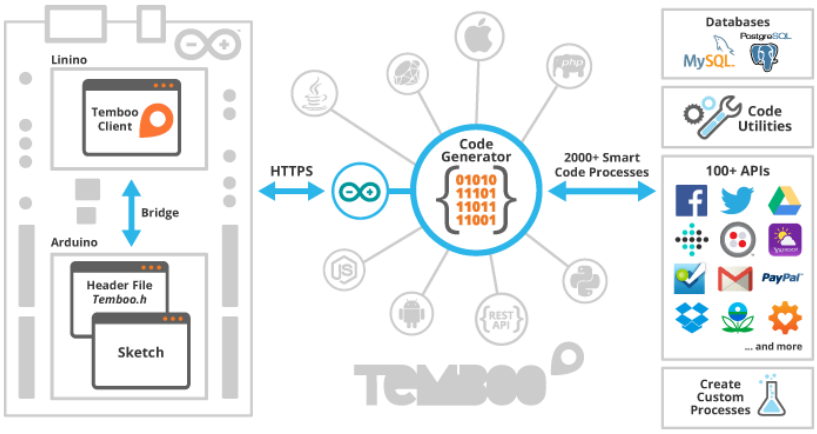


Figura 1.3: Ejemplo de plataforma IoT Temboo. Fuente: `https://aprendiendoarduino.wordpress.com/tag/temboo/`

cuenta con una herramienta para generar archivos CSV en caso de que se interrumpa la conexión a internet, para así perder la menor cantidad de datos posible. También contará con la visualización de los datos en la misma nube, para ello emplearemos la plataforma Ubidots, que nos permite generar gráficas en la misma nube, además de envío de correos en caso de que fuese necesario, establecimiento de condiciones, entre otras muchas posibilidades. Debido a la capacidad de toma de decisiones y activación de eventos en la nube, se podría tratar del sistema de tratamiento energético de una casa inteligente.

El lenguaje de programación empleado será Python en gran medida, debido a la comodidad que nos da y la gran cantidad de librerías con las que cuenta, además de la comunidad que posee de cara a buscar ejemplos de código y ayudas.

1.3 Explicación de la estructura del trabajo.

Este TFG consta de siete capítulos diferenciados, entre los que se incluye esta ligera introducción. En el capítulo 2 o "Marco teórico", será donde se comentará la historia de las Smart Grids, mencionando las primeras y cómo han ido evolucionando con el paso de los años, para finalmente acabar hablando de las Casas Inteligentes o Smart Homes, mencionando también las tecnologías empleadas tanto en sus primeros años como en la actualidad. En el capítulo 3, Objetivos, será donde se mencionará desde qué punto comenzamos a trabajar y cuáles son los objetivos que queremos alcanzar al finalizar este trabajo. En el capítulo 4, Metodología, hablaremos de las tecnologías que se han empleado y de las distintas fases por las que hemos ido avanzando. El capítulo 5, Desarrollo, tratará de manera más específica las fases que se mencionarán en el capítulo anterior, añadiendo los códigos que se han realizado. Durante el capítulo 6, llamado Resultados, mencionaremos cuáles han sido los resultados finales del trabajo, comparándolo con los objetivos que se tenían. Finalmente se mencionarán algunas conclusiones en el capítulo 7, para después dejar paso a la bibliografía.

2 Marco Teórico.

2.1 Definición de REI.

Comenzaremos definiendo qué es una Smart Grid (red eléctrica inteligente o REI), siendo ésta una red eléctrica que incorpora tecnología para que haya una comunicación fluida en ambas direcciones (instalación y usuario), haciéndola, de esta manera, inteligente. Se emplean herramientas informáticas y domóticas. Se pretende buscar la máxima eficiencia energética. [2]

Haciendo referencia al funcionamiento, podríamos simplificarlo diciendo que en una Smart Grid los circuitos "hablan" y con capacidad de monitorizarse a sí misma. Lo que se pretende conseguir es tener una información muy detallada sobre el consumo. Con tal de mantener la seguridad del suministro en un sistema eléctrico, se están impulsando mucho las iniciativas de redes inteligentes, y con ellas se anticipan soluciones en el ámbito de las nuevas tecnologías de almacenamiento, monitorización, autoconsumo, vehículos eléctricos… En España se han trabajado 23 proyectos desde el año 2015, de los cuales 10 han dado como resultado nuevos procesos que se han incorporado en la operación del sistema.[2]

Las primeras mejoras en redes inteligentes comenzaron con el uso de electrónica de control, medición y monitoreo. No fue hasta 1980 cuando se comenzó a emplear la lectura automática de contadores para el seguimiento del consumo de grandes clientes, que no se instauró hasta 1990. En el año 2000, Telegestore, un proyecto italiano, instauró contadores inteligentes en aproximadamente 27 millones de hogares, los cuales se conectaron a través de bajo ancho de banda por la línea de energía, lo que en otros experimentos es conocido como el término banda ancha sobre líneas eléctricas o BPL. Otros proyectos emplearon conexiones inalámbricas, además de añadir medición de otros servicios, como podría ser gas o agua. Fue en 1990 cuando Bonneville Power Administration amplió su investigación en redes inteligentes, con el uso de sensores que eran capaces de encontrar anomalías en la red eléctrica en grandes áreas. Esta investigación acabó con el surgimiento del Sistema de Medición de primera operativa de área extensa o WAMS en el año 2000. Se destaca también la red de malla de Austin, Texas, que está presente desde 2003, y la red inteligente en Boulder, Colorado, presente desde 2008.[3]

La aparición del concepto de Smart Grid se debe a la necesidad de ahorro energético y al uso de energías renovables, y también a la optimización del negocio y a la necesidad de la mejora de la eficiencia en los sistemas. Esto es debido en gran parte al Strategic Energy Technology Plan (SET Plan), un plan lanzado por la Unión Europea para conseguir los objetivos 2020, entre los que está el incremento de la eficiencia energética, reducción de emisión de gases de efecto invernadero, aumento del uso de energías renovable; y cuyo objetivo prioritario es el desarrollo de redes eléctricas inteligentes. [4] El desarrollo de redes eléctricas inteligentes implica el desarrollo de dos nuevos conceptos, tales como [5]:

- Telegestión: Medida y gestión a distancia y en tiempo real de los consumos del usuario final. Es una de las capacidades más necesarias para una Smart Grid, ya que permite

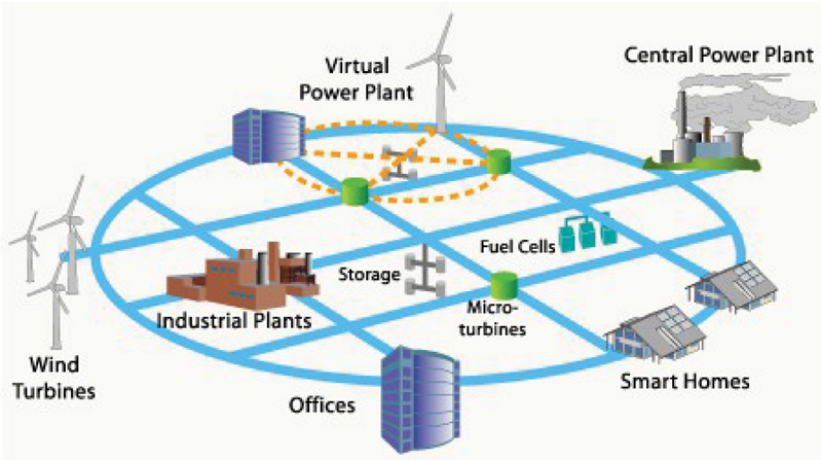


Figura 2.1: Principio de funcionamiento tecnológico para una red inteligente. Fuente: `https://www.voltimum.es/articulos-tecnicos/principio-funcionamiento`

conocer los hábitos de consumo de un cliente, pudiendo así ofrecerle ofertas de consumo o tarifas que se adapten más a sus necesidades.

- Gestión activa de la demanda: Consiste en optimizar el consumo de un cliente en función del comportamiento que hayamos observado gracias a la telegestión.

Entre los beneficios asociados a la instauración de una REI están:

- Transmisión más eficiente de la electricidad
- Un restablecimiento más rápido y eficaz tras una interrupción
- Reducción de costes, debido a la reducción de picos de demanda
- Instauración de energías renovables
- Seguridad

2.2 Las comunicaciones en una REI.

Aunque las Smart Grids están todavía en continuo desarrollo y quede mucho camino por avanzar, es necesario el desarrollo y la mejoría en este camino, con tal de garantizar la eficiencia energética y el ahorro para conseguir un mundo sostenible. Existen estudios que han dado datos como que si instaurásemos redes eléctricas tan solo un 5 por ciento más eficientes, el ahorro equivaldría a la emisión de 53M de coches. Es obvio que el desarrollo de estas Smart Grid implica mejorías en el apartado de comunicaciones, con tal de permitir una mejor fluidez de la información, menor pérdida de ésta, velocidad de transmisión, entre otras. Se definen diversas generalidades a la hora de hablar de las comunicaciones en Smart Grid. Por un lado hablaremos del GIS (Sistema de información geográfica), cuya calidad debe der ser excepcional, ya que si el sistema fallo puede ocasionar cortes o accidentes. Haciendo

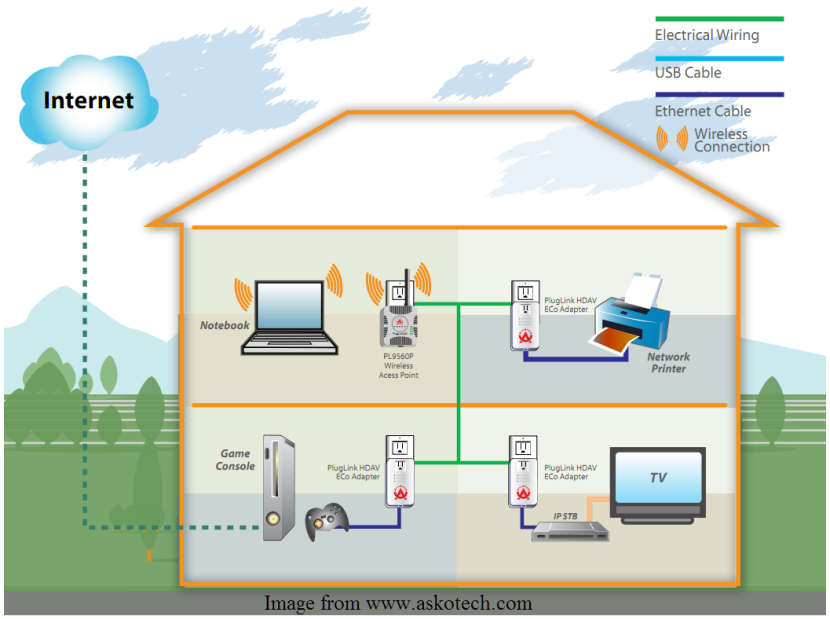


Figura 2.2: Ejemplo de Home Area Network. Fuente: `www.askotech.com`

referencia a los SCADA, puede existir un HMI (Human Machine Interface) instalado sobre un servidor para controlar los procesos desde la pantalla de monitorización. El SCADA será el encargado de realizar análisis en tiempo real o estudios posteriores para generar indicadores que permitirán realimentar al sistema.[6] Son necesarias tres redes de comunicaciones[6]:

- Home Area Network: es un conjunto de dispositivos de diferentes clases como electrodomésticos, ordenadores personales... que están instalados en un hogar y que se encuentran interconectados entre sí. Muchos de ellos pueden ser operados a distancia por Internet mediante una app o una web por ejemplo.
- Building Area Network: se trata de una red local que cubre un edificio entero. Puede ser un conjunto de Building Area Network más pequeños.
- Industrial Area Network: basado en usar Ethernet en un ámbito industrial junto con otros protocolos de comunicación para ayudar al control en tiempo real.

También se requerirá la implantación de micro redes, dispositivos locales de almacenamiento y cargas controlables. Se destacan las siguientes redes de comunicación que permiten diversos beneficios al usuario, entre ellos el monitoreo del uso que le dan a la energía o economizar en función del precio de la energía. Estos son[6]:

- WiMax [1]: se emplea sobre todo para interconectar a los medidores inteligentes.
- 3G: para control y monitoreo.

[1] WiMax es un protocolo de comunicación de datos que trabaja en ondas de radio de 2.5 Gz a 5.8 GHz y que puede tener una cobertura de hasta 70 km.

- MPLS: para conectar redes que emplean diferentes tecnologías.
- ZigBee: enfocado para comunicación remota entre medidores, incluidos también gas y agua.
- Home Plug [2]: interconexión con bombas de agua, audio, televisión, nevera, vehículos eléctricos…
- WiFi y 4G LTE: se emplean para obtener datos desde prácticamente cualquier dispositivo electrónico, aunque el 4G LTE puede ser usado para control y monitoreo en alto y medio voltaje y conexión con hogares, ya que no es recomendable emplearlo en sectores rurales debido al empleo de altas frecuencias.

Existen más tecnologías de comunicación, como por ejemplo la fibra óptica, que es la más fiable, puesto que nos da una gran seguridad y además es inmune a interferencias electromagnéticas, además de la gran flexibilidad que nos aporta. El único punto negativo es su elevado coste, por lo cual tan solo haremos mención de ella en este proyecto.

2.3 Definición de Edificio Inteligente.

Las Smart Grids han generado un gran impacto en el desarrollo de edificios inteligentes, tal y como se revela en un estudio del Consejo de Edificios Inteligentes e Integrados (IIBC en inglés) titulado "2011 Landmark Research". Los edificios inteligentes se caracterizan por ser edificios equipados con sistemas de automatización y control de la iluminación, calefacción, refrigeración… Entre otras cosas, los cuales están interconectados entre ellos y conectados a internet. El objetivo de los edificios inteligentes es satisfacer aspectos como confort o seguridad, y abarcan cualquier tipología, ya sean viviendas, oficinas, industrias, u otras cosas. Los edificios inteligentes no son solo labor de arquitectos, también intervienen personas como ingenieros en telecomunicaciones, industriales, y otros técnicos. En conclusión, consiste en combinar las nuevas tecnologías de la información y los sistemas de automatización en proyectos constructivos. El desarrollo de estos edificios inteligentes debe cumplir cuatro objetivos [7]:

- Nivel arquitectónico: a pesar de ser un edificio inteligente, debe de cumplir funcionalidades de confort, operatividad y durabilidad que debe de tener cualquier edificio corriente. Además debe de tener un diseño estético.
- Nivel tecnológico: el edificio debe de contar con servicios integrados y con sistemas de comunicación avanzados.
- Nivel ambiental: se deben de cumplir una serie de compromisos respetuosos con el medio ambiente, que abarcan desde los materiales abarcados en la construcción, optimización de los elementos de iluminación, eficiencia energética…
- Nivel económico: obviamente debe conseguir reducir todos sus costes de funcionamiento y mantenimiento, al mismo tiempo que se consigue alargar su vida útil.

[2] Home Plug es una familia de especificaciones que se basa en comunicación por red eléctrica, empleando conexiones eléctricas ya existentes en el hogar.

2.4 Funcionamiento de un Edificio Inteligente.

Una de las mayores es preguntas que se suele hacer es: ¿cómo se dota a un edificio de inteligencia? Esto básicamente se consigue incorporando sensores, captadores y detectores que transmiten información a la unidad de proceso, donde se tomarán decisiones y enviando información a los actuadores en función de los datos obtenidos. Además se permite a los usuarios de edificios enviar órdenes para activar diversos equipos a distancia. Se puede catalogar la inteligencia de un edificio en nivel de la tecnología aplicada, así como la cantidad y calidad de los sistemas automatizados. Estos niveles son [7]:

- Nivel 1: corresponde a edificios con ciertos servicios de telecomunicaciones avanzadas, pero que no interactúan con instalaciones y equipos.
- Nivel 2: este sistema ya cuenta con sistemas de automatización y control, pero no están interconectados.
- Nivel 3: son edificios completamente inteligentes que cuentan con una total integración entre la actividad, automatización y telecomunicaciones.

La estructura básica de un edificio inteligente es:

- Sistema básico de control del edifico y su entorno, que se encarga de verificar el estado de las instalaciones básicas de suministro, como electricidad, agua, gas o ascensores y escaleras mecánicas.
- Sistema de control de seguridad del edificio: actúa para proteger personas, equipación e información. Estos sistemas son detección de fuegos, fugas de agua, humo, circuitos de vigilancia…
- Sistemas de ahorro de energía en el edificio, que transmiten temperatura, estudia consumos, programa horarios de iluminación y climatización, optimizando el funcionamiento de sistemas. Este trabajo de fin de grado estará estrechamente relacionado con este ámbito.

En España todavía no está muy desarrollado el ámbito de los edificios inteligentes, a pesar de eso, existen empresas que trabajan en estos ámbitos, como pueden ser:

- Hermespro Edificios Inteligentes SL
- Proyectos Y Desarrollo de Edificios Inteligentes SL

2.5 Tecnologías empleadas en un Edificio Inteligente.

Uno de los principales objetivos a la hora de diseñar un edificio inteligente es la gestión de la energía, y más concretamente los relacionados con la iluminación y la climatización. Tras los servicios de iluminación y climatización, el más demandado es la seguridad. En el mercado existen una gran cantidad de opciones para optimizar el uso de la energía y así reducir su consumo, pero no es tan simple como puede parecer, ya que para cada tecnología

Figura 2.3: Estructura básica de un edificio inteligente.

que instauramos en un inmueble necesitamos un tipo de control distinto. Un ejemplo de esto es que para soluciones de climatización suelen emplearse soluciones LON o BACNet, mientras para optimización de la energía suele emplearse Modbus.

- LON: se trata de una solución para sistemas de control. Es la base para un sistema abierto e interoperable en el que los productos y soluciones de empresas líderes del mundo se reúnen en una implementación simple y sencilla que integra varios componentes del sistema en una solución completa. [8]

- BACNet: es un protocolo de comunicación de datos diseñado para comunicar entre sí a los diferentes aparatos electrónicos presentes en los edificios actuales (alarmas, sensores de paso, aire acondicionado, calefactores, etc.). [9]

A pesar del problema de compatibilidad que podríamos tener, existen tecnologías multiprotocolo que solucionan este problema, como el SmartStruxure de Schneider Electric, el cual permite, mediante una misma herramienta, supervisar todos los sistemas. Este sistema funciona con el software SmartStruxure.

Otro tipo de tecnologías que se añaden a un edificio inteligente son seguridad, control de acceso o cierre de circuitos. Gracias al desarrollo en sensores y detectores, no se necesita una obra intrusiva, ya que pueden colocarse sobre paredes, puertas, entre otros. Un gran ejemplo de tecnología de las que se dispone a día de hoy en el mercado es el software Smart Things de Samsung, el cuál nos permite conectar dispositivos domésticos. Permite realizar acciones como apagar la luz desde la tele e incluso nos sugiere cómo podríamos aprovechar de manera más eficiente la energía. Está formado por un Smart Hub, que sería el ”cerebro”, y diversos sensores que controlan presencia, alimentación... [10]

Figura 2.4: SmartStruxure de Schneider Electric. Fuente: `http://squaredeal-usa.com/cart/schneider-electric-SXWASB24X10001-smartx-controller-as-b-24-sxwasb24x10001-24-i-o-bac html`

Figura 2.5: Sensores con los que cuenta Smart Things. Fuente: `https://www.smartthings.com/products/-/filter/categories/sensors`

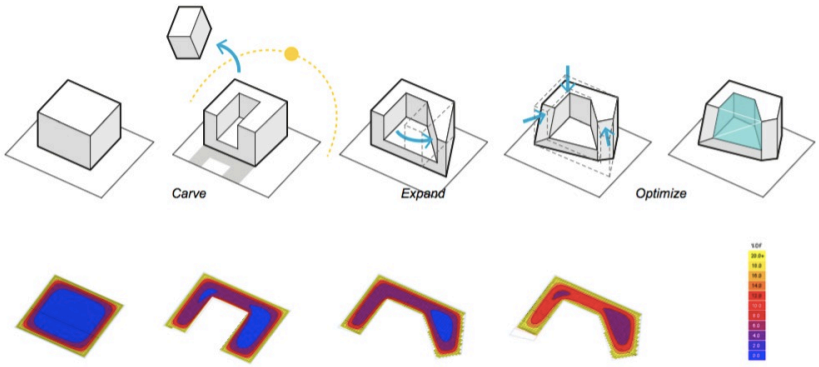


Figura 2.6: Disposición del atrio del The Edge. Fuente: `https://www.archdaily.pe/pe/790319/the-edge-plp-architecture`

2.6 Ejemplos de edificios inteligentes.

Destacamos el The Edge, situado en Ámsterdam, el The Crystal, situado en Londres y la Torre Europa situada en Madrid.

2.6.1 The Edge.

Se trata de las oficinas de Deolitte en Ámsterdam. Cuenta con una App propia que conecta a cada uno de los trabajadores con el edificio, de manera que el mismo edificio sabe qué reuniones tienen hoy, cuánto tiempo pasarán en el edificio e incluso qué coche tienen. El edificio se reorganiza de manera inteligente, de manera que nadie tiene un despacho fijo, si no que el mismo edificio asigna salas a cada trabajador en función de sus preferencias, incluyendo preferencias de iluminación, temperatura, humedad... Gracias a esto el edifico es capaz de llegar a cerrar plantas enteras que considera que no serán usadas ese día, suponiendo un gran ahorro. [11] No solo es una obra maestra en cuanto a tecnología, si no también arquitectónicamente hablando, ya que el hecho de que esté compuesto por cristaleras inteligentes, que se mueven para dar la mayor cantidad de luz natural posible. El atrio es el pulmón del edificio, la ventilación del espacio de la oficina al mismo tiempo que proporciona una memoria intermedia con el exterior de una manera que reduce el uso de energía tanto en verano como en invierno. Así como su control de la temperatura neutra la energía, el diseño eficiente de la energía y la tecnología de generación de energía verde, The Edge captura agua de lluvia y la almacena en cisternas para su uso en los inodoros y el riego de plantas en los jardines interiores y exteriores. [12]

2.6.2 The Crystal.

Es un edificio inteligente situado en Newham, Londres, y que lleva en funcionamiento desde septiembre de 2012. Cuenta con autoabastecimiento energético mediante placas solares y aprovechamiento del calor del entorno. Las principales cualidades con las que cuenta el edificio son [13]:

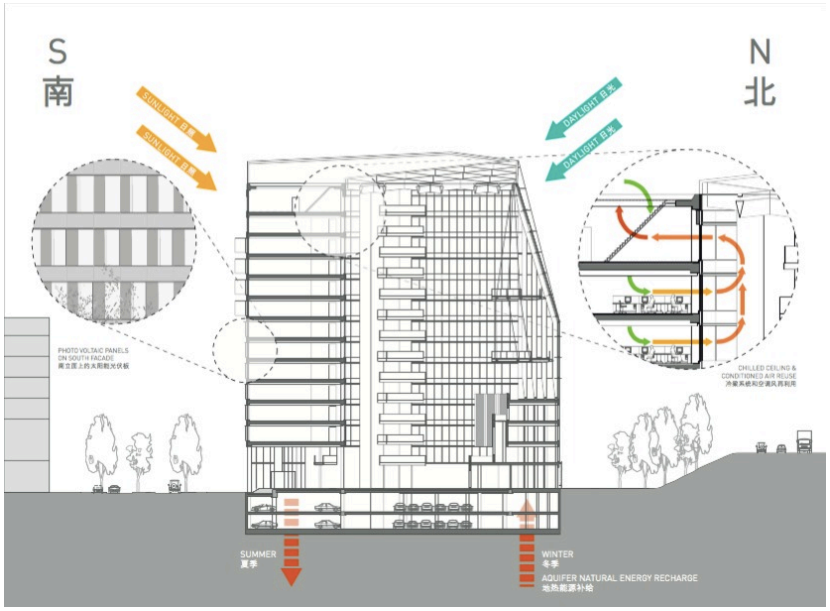


Figura 2.7: Sistemas de ventilación en The Edge. Fuente: `https://www.archdaily.co/co/790319/the-edge-plp-architecture`

Figura 2.8: Edificio The Crystal. Fuente: `https://www.thecrystal.org/`

- Control Remoto de instalaciones: el edificio es capaz de distribuir la energía de forma eficiente, almacenándola en una pila para después emplearlo en las horas puntas.
- Empleo de la luz natural gracias a estar completamente acristalado.
- Cuenta con rejillas motorizadas para aprovechar al máximo la ventilación que proviene del exterior.

Hablamos de un edificio que consume 70kWh por metro cuadrado, lo que viene a ser un 25 por ciento menos de lo que establece la legislación de energía. El edificio fue diseñado por Siemens y obviamente está automatizado con tecnología de la marca, contando con automatización por parte de la tecnología Desigo. Por otro lado el Centro de Operaciones Avanzadas en Frankfurt (Alemania) se encarga de mantenerlo de manera remota. Desigo es un sistema de gestión de edificios que permite controlar, regular y optimizar la calefacción, ventilación, aire acondicionado, protección contra incendios, control de seguridad, entre otros. El hecho de ser

Figura 2.9: Esquema general del funcionamiento de Desigo. Fuente: `https://new.siemens.com/es/es/productos/building-technologies/automatizacion/desigo.html`

una tecnología modular le permite ser fácilmente ampliable.[14]

2.6.3 Torre Europa.

La Torre Europa es un edificio situado en Madrid, y que está presente desde 1985, pero no comenzó a ser un edificio inteligente hasta que se realizó una reforma que comenzó en 2015. Una de las principales características añadidas en esta reforma, y que al mismo tiempo la convirtió en un edificio inteligente fue el sistema Power-over-Ethernet (PoE), por parte de Philips Lighting, que permite conectar la luz a los sistemas informáticos del edificio. El sistema recoge datos de consumo anónimos, lo que permite ajustar niveles de iluminación, calefacción, entre otros; y que a su vez asegura que, por ejemplo, cuando todos los usuarios salgan de una sala se apague la luz. Este uso eficiente de la iluminación llega a permitir ahorros de un 70 por ciento, que equivalen a unas 15 toneladas de CO2 por planta. [15] Esta tecnología llamada Power-over-Ethernet, es una tecnología que incorpora alimentación eléctrica a una infraestructura LAN estándar. Permite que la alimentación eléctrica se suministre a un dispositivo de red (switch, punto de acceso, router, teléfono o cámara IP, etc) usando el mismo cable que se utiliza para la conexión de red. El PoE se rige bajo las normas del estándar IEEE 802.3af. Está compuesto de cuatro fases [16]:

- Primer bloque: “Polarity Protection” o “Auto-polarity Circuit”. La tensión puede venir mediante Ethernet o mediante otros pares alternativos de tensión.

- Segundo bloque: “Signature and Class circuitry”. Sirve para no aplicar tensión a un dispositivo que no implementa PoE. Para realizar la comprobación, se realizan 4 etapas, aplicando primero una tensión baja, para buscar una resistencia de 25 kiloohmios, de manera que si es muy alta o muy baja no hará nada, pudiendo proteger a un dispositivo no PoE de uno que sí. Las fases 3 y 4 se aplicarían en caso de ser PoE, de manera que si es PoE buscará qué alimentación requiere y en la fase 4 el dispositivo sabrá cuál es la alimentación máxima para que funcione el sistema.

- Tercer bloque: "Control Stage". El controlador solo se enciende cuando el voltaje es 35 V.
- Cuarto bloque: "Convertidor DC/DC".

Power-over-Ethernet es una tecnología que cuenta con muchas ventajas, como la poca pérdida de tiempo y de dinero que supone su instalación, la capacidad de controlar circuitos de manera remota, la poca necesidad de cables para su instalación; que supone un ahorro de espacio. Además, PoE permite al usuario conectar a todos los sistemas basados en PoE al Sistema de Alimentación Interumpida (SAI) [3], lo que supone una gran ventaja en caso de que se caiga la red eléctrica. [16]

La única desventaja que encontramos en Power-Over-Ethernet es la poca diversidad de protocolos a la hora de realizar la interconexión entre equipos.

Estas integraciones en la reforma son las que han galardonado a la Torre Europa con premios como el certificado LEED Gold, que supone que el edificio cuenta con prácticas sostenibles, además de haber conseguido el premio a la mejor reforma del año 2019, por parte del Consejo de Edificios Altos y Hábitat Urbano.

2.7 Sistemas embebidos para tecnologías Smart Home y gestión energética.

Existe una gran variedad de sistemas embebidos en el mercado, desde los sistemas embebidos industriales hasta los sistemas TIC. Por un lado, los sistemas embebidos industriales, como su nombre indica, presentan unas características enfocadas a un entorno industrial, y a tareas con una mayor carga de trabajo, como puede ser controlar una cadena de montaje en una fábrica. Dentro de esta clasificación podríamos incluir a los autómatas IoT, como por ejemplo el sistema embebido de Siemens SIMATIC IOT2040. Entre sus características [17]:

- CPU: Intel Quark x1020 (x86 400 MHz) con funciones de seguridad.
- Memoria del sistema: 1 GB, 8 FLASH MB, 256 kB.
- Tipo de memoria del sistema: DDR3 RAM, SRAM.
- Interfaces de medios: USB 2.0, USB Host and Client.

Podemos observar que posee buena calidad de hardware, y cuenta con un precio bastante económico, entre 200 y 300 euros en función de dónde se adquiera. Aquí entra una de las principales razones por las que he seleccionado un sistema TIC, ya que la intención de este TFG es el diseñar un prototipo contando con tecnologías mucho más económicas. En cuanto a los sistemas TIC, se refiere a la preparación que tienen estudiantes para satisfacer las necesidades de tecnologías en cómputo y organización [18]. Un gran ejemplo de estos sistemas es el sistema embebido Raspberry Pi 3, que es el empleado para este TFG, el cual comparando su precio con el Simatic que he mostrado cuesta aproximadamente 30 euros, una de las

[3]Sistema que gracias a sus baterías u otros elementos, permite proporcionar energía eléctrica a los dispositivos en caso de apagón.

Figura 2.10: Sistema embebido industrial Simatic de Siemens.

grandes razones por las que se suele elegir este tipo de sistemas para trabajos realizados por estudiantes. Además, si comparamos la calidad del hardware con el que cuenta el sistema, tampoco se queda corto [19]:

- CPU: SoC Broadcom BCM2835 de un núcleo (ARM11).
- Memoria: 512 MB.
- Conexiones: 40 conexiones GPIO.
- USB: 4 puertos USB 2.0.

Además sistemas como Raspberry, que son tan conocidos mundialmente, cuentan con una gran comunidad detrás que hace proyectos con intenciones académicas, lo que a personas que se hayan en casos como el mío nos beneficia muchísimo, cosa que con sistemas industriales no sucede.

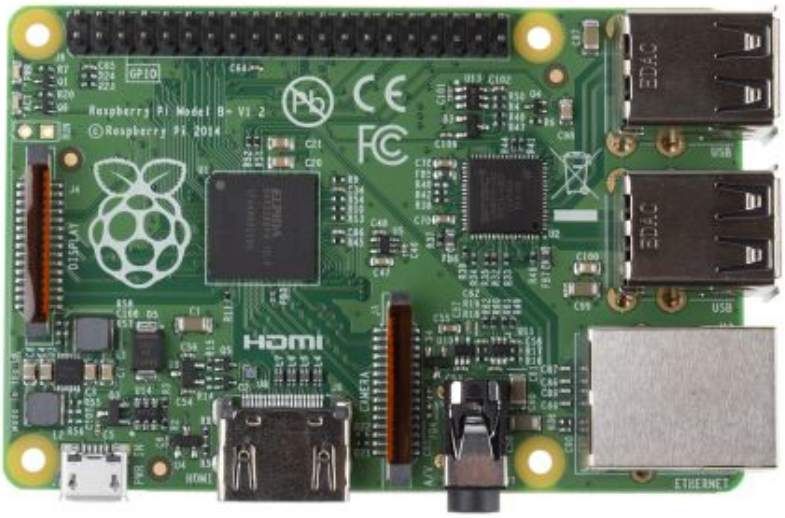


Figura 2.11: Sistema embebido TIC Raspberry Pi.

3 Objetivos.

3.1 Objetivos generales.

El objetivo principal de este TFG es el conseguir un sistema que, mediante la lectura de datos de consumo, sea capaz de conseguir optimizar el gasto energético de manera autónoma, además de tratarlos en la nube.

3.2 Objetivos específicos.

Una vez mencionados los objetivos de manera más general, paso a mencionar los objetivos más específicamente, estos son:

- Lectura de datos de una red trifásica empleando SPI con el sistema embebido.
- Comunicación total entre la nube y el sistema embebido, empleando HTTP y la librería Requests de Python. Esta comunicación será en ambos sentidos, tanto del sistema embebido a nube como de nube a sistema embebido.
- En la nube, empleando Ubidots, exprimir al máximo su potencial, lo que incluye:
  - Empleo de eventos dentro de la nube.
  - Graficar los datos.
  - Empleo de widgets.
- Generar archivos CSV en caso de que no exista conexión a internet, para comprobar esto emplearemos Ping, que sirve para comprobar si recibimos respuesta de una IP.
- Subir los archivos en formato estándar a la nube cuando se recupere la conexión a internet, para evitar almacenar muchos datos a nivel local.
- Manejo de fechas de los datos, para así tener constancia de cuándo se leyeron los datos.

4 Metodología

Desde la sección Tipos de metodologías de diseño software hasta Arquitectura software se recomienda leer [20, 21]

4.1 Tipos de metodologías de diseño software.

La metodología de desarrollo software es un apartado de trabajo donde se estructura y planifica cómo va a desarrollarse un sistema software. Con el paso de los años, han surgido una gran cantidad de metodologías. Antes de mencionar las metodologías más típicas, hay que definir qué es un método y qué es una metodología. Un método es un conjunto de herramientas que, aplicadas de manera correcta, nos permiten alcanzar un objetivo. Por otro lado, una metodología es el marco en el cual se estructura el desarrollo de un sistema. Entre las muchas metodologías se destacan:

- Cascada.
- Prototipos.
- Incremental.
- Espiral.
- Desarrollo rápido de aplicaciones (RAD en inglés).
- Metodología de programación extrema (XP en inglés).

4.2 Cascada.

Se trata de un proceso secuencial. Las etapas características son las fases de análisis de necesidades, el diseño, implantación, pruebas, integración y mantenimiento. Es una de las metodologías en las que se hace más hincapié en planificación, horarios, fechas, presupuestos y ejecución, además de que supone un estricto control durante la vida del proyecto.

En este modelo cada etapa es una unidad de desarrollo con un pequeño descanso en el medio, de manera que cada nueva etapa inicia en el momento en el que la otra etapa termina; y esos descansos quedan para la confirmación por parte del cliente. Es la metodología más tradicional.

4.3 Prototipos.

Destaca por la posibilidad de desarrollar modelos que permiten ver una funcionalidad básica sin incluir todo lo que requiere el modelo ya terminado. Permite que el cliente pueda evaluar

Figura 4.1: Esquema básico de la metodología cascada. Fuente: `https://www.yunbitsoftware.com/blog/2016/05/20/desarrolo-de-software-metodologias-waterfall-agile/`

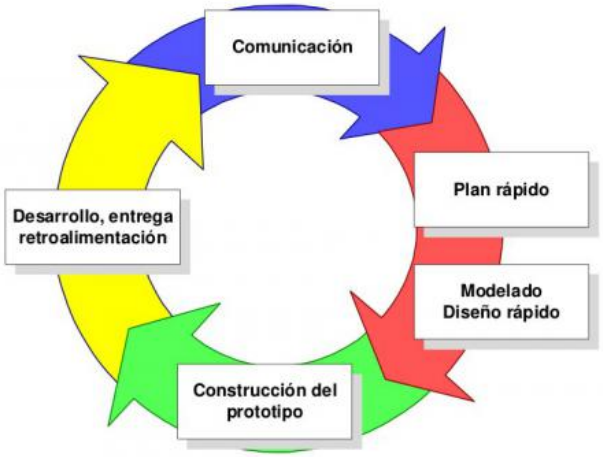


Figura 4.2: Esquema básico de la metodología prototipos. Fuente: `http://marich.blogspot.es/1459397526/metodologia-de-prototipos/`

de manera temprana el producto, para saber si se cumplen las expectativas que esperaba. Lo mejor de esta metodología es que tiende a resolver problemas de diversificación que surgen en la metodología de cascada.

Sirve sobretodo para dar una idea clara del proceso funcional del software.

4.4 Incremental.

Es una estrategia para controlar la complejidad y los riesgos, desarrollando una parte y dejando el resto para el futuro. Las fases son:

- Inicialización
- Periodos de iteración: se realiza una primera iteración con las características iniciales, para que al final de la primera iteración quede un pequeño prototipo, para dar inicio a una siguiente iteración, que estará destinada al análisis.
- Lista de Control: tras cada iteración, se realiza un control de la misma.

Cada una de las fases generará un prototipo que cada vez estará más evolucionado.

4.5 Espiral.

Se basa en dividir el proyecto en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo. En cada viaje que hagamos por la espiral pasamos por:

- Determinar objetivos, alternativas y desencadenantes de la iteración.
- Evaluación de alternativas e identificar y resolver los riesgos.

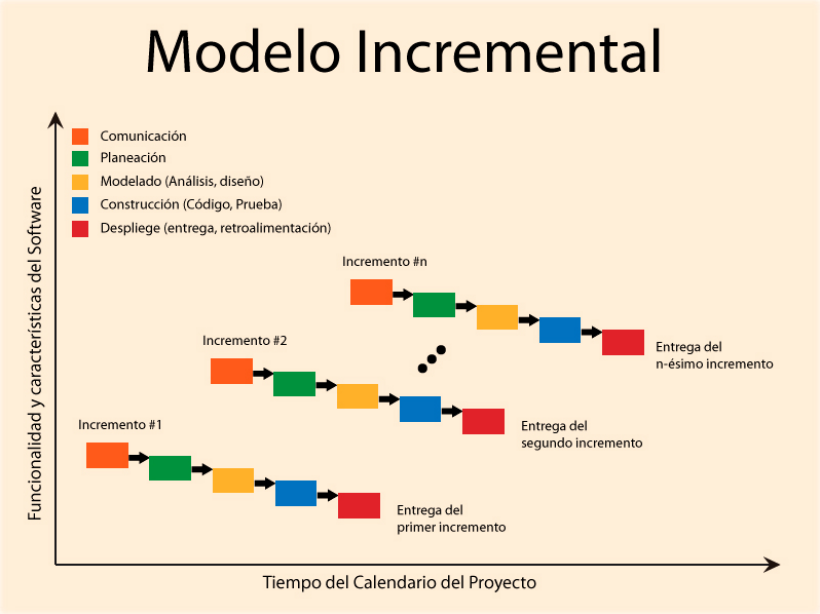


Figura 4.3: Esquema básico de la metodología incremental. Fuente: `http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html`

- Desarrollo y verificación de los resultados de la iteración.
- Plan de próxima iteración.

Suele emplearse en proyectos grandes y ambiciosos como puede ser un sistema operativo.

4.6 RAD.

Es una metodología de desarrollo de software que implica, por un lado el desarrollo iterativo y por otro la construcción de prototipos. El principal objetivo es conseguir un desarrollo rápido y una entrega de alta calidad, reduciendo los riesgos inherentes partiéndolo en segmentos más pequeños.

Está basado en el uso de iteraciones y en el manejo de prototipos, pero la principal diferencia que ofrece esta metodología es el uso de herramientas CASE para acelerar el proceso.

4.7 XP.

Es una metodología que se emplea para evitar el desarrollo de funciones que no son necesarias. El principal inconveniente es que sus métodos peculiares pueden requerir más tiempo y recursos.

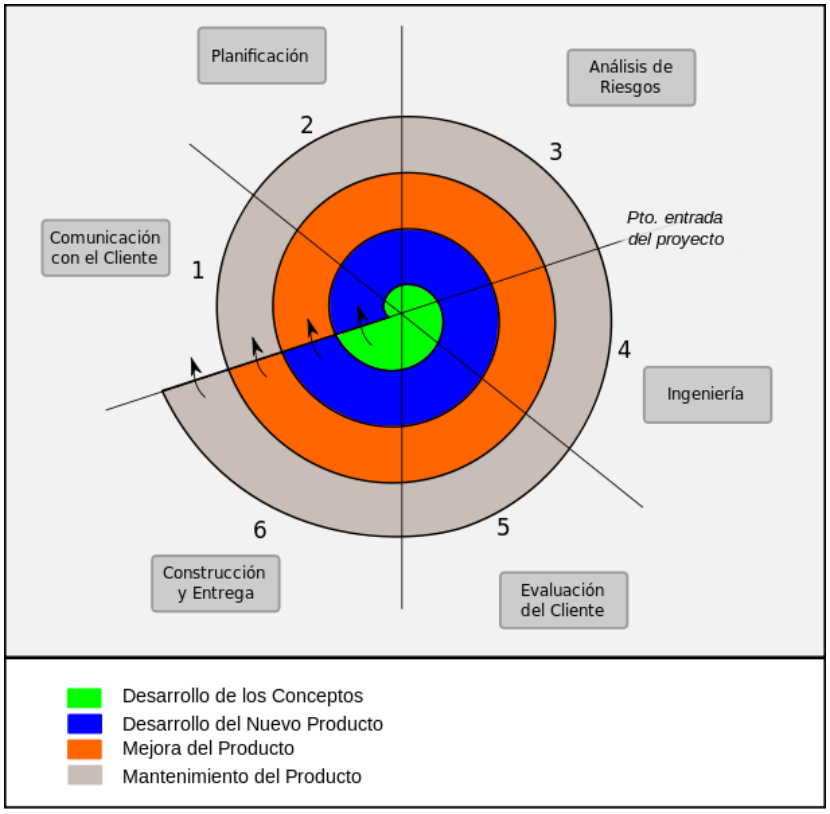


Figura 4.4: Esquema básico de la metodología espiral. Fuente: `http://sofware1nathalygrijalva.blogspot.com/2012/10/modelo-espiral.html`

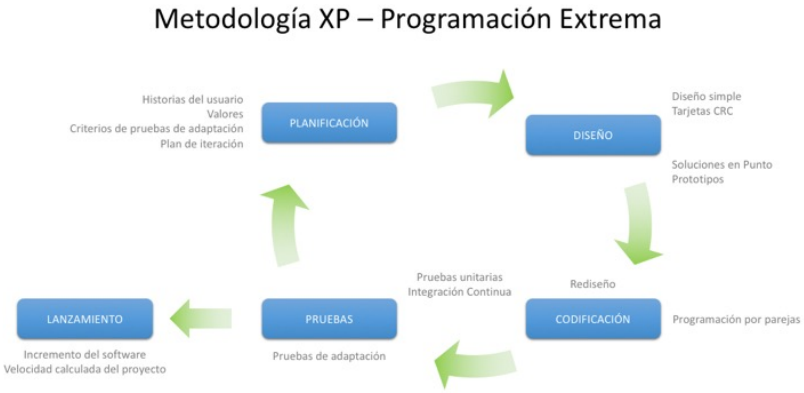


Figura 4.5: Esquema básico de la metodología XP. Fuente: `http://www.diegocalvo.es/metodologia-xp-programacion-extrema-metodologia-agil/`

4.8 Selección de metodología de diseño.

Una vez mencionadas y caracterizadas cada una de las metodologías (a pesar de que existen muchas más), hay que seleccionar qué tipo de metodología será la empleada para este TFG. Como bien he mencionado en los capítulos anteriores, la finalidad de este TFG es crear un prototipo de sistema embebido capaz de comunicarse completamente con la nube y almacenar datos de manera local en caso de una caída de internet, a grandes rasgos.

Las metodologías que más se adecuan a este TFG serían:

- Prototipos: por la capacidad que nos da de presentar un modelo que no contiene todas las características finales, y de la funcionalidad básica que queremos darle al sistema.
- Incremental: es una metodología adecuada para este trabajo ya que nos permite ir añadiendo mejoras de manera progresiva al sistema, para después comenzar una etapa de análisis, que en este caso sería una corrección por parte del tutor.
- Espiral: nos ofrece la posibilidad de ir planteando nuevas mejoras de manera continua, y, a pesar de estar destinada a proyectos más grandes y ambiciosos, encajaría muy bien en este proyecto.

Finalmente, la metodología más adecuada considero que es la metodología incremental, en la cual podríamos considerar que en cada iteración se plantea una nueva funcionalidad, para después integrarla y corregirla con el tutor, de manera que a partir de esa corrección surge una nueva iteración, en base a ideas que han podido surgir en incrementos anteriores. Tras cada incremento o iteración obtendríamos un sistema con una nueva funcionalidad. Por ejemplo, en un incremento se plantearía cómo realizar la comunicación con la nube, y el prototipo final de esa iteración sería el sistema embebido comunicándose con la nube mediante HTTP, y la siguiente iteración comenzaría añadiendo funcionalidades en la nube.

4.9 Arquitectura software.

Una arquitectura software indica la estructura y cómo van a interaccionar cada una de las herramientas software que se aplican en un proyecto. [22]

En proyectos que se emplean tecnologías IoT observamos un patrón similar en las arquitecturas software, tal y como se observa en la figura 4.6 [23].

Se puede observar que en la primera capa están los dispositivos, también conocidos como las cosas, en la siguiente capa las comunicaciones, que dan paso a los eventos y análisis de los datos para finalmente mostrarse en un dashboard, emplearse en alguna página web o una API.

De manera más específica, la arquitectura software que presenta este TFG está formada por, que se puede observar en la figura 4.7:

- Capa de dispositivos/cosas: En esta capa se encontraría el sistema embebido.
- Capa de comunicaciones: donde se hallaría HTTP, que es nuestro protocolo de comunicaciones con la nube, y SPI, nuestro bus de datos para comunicar el ADC con el sistema embebido.

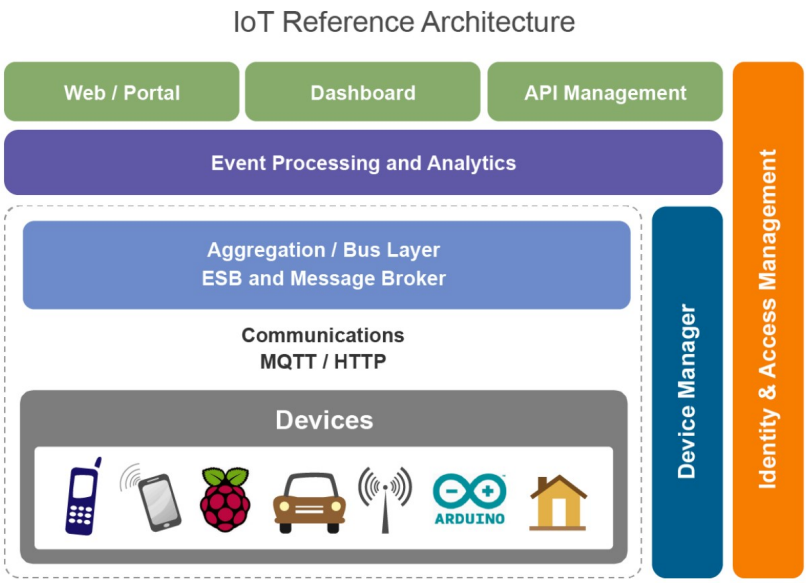


Figura 4.6: Este suele ser el esquema básico de un proyecto que emplea tecnologías IoT. Fuente: `https://blogs.deusto.es/master-informatica/iot-ya-esta-aqui/`

- Capa cloud: donde se encontraría la plataforma IoT Ubidots.
- Capa de visualización: sería la capa final, donde se encontrarían los eventos o los múltiples dashboards para visualizar los datos.

4.10 Herramientas software.

Las herramientas software empleadas han sido:

- Geany: es un editor de texto pequeño y ligero basado en Scintilla con características básicas de entorno de desarrollo integrado [24]. Ha sido empleado para la programación del algoritmo.
- TeXstudio: es un editor de LaTeX de código abierto y Multiplataforma con una interfaz similar a Texmaker [25]. Se ha empleado para la redacción de esta memoria.
- Microsoft Word: editor de texto creado por Microsoft. Se ha usado como plantilla previa para la redacción de la memoria.
- Ubidots: plataforma IoT que nos permite manejar nuestra propia nube, ofreciendo herramientas de graficado, de control...

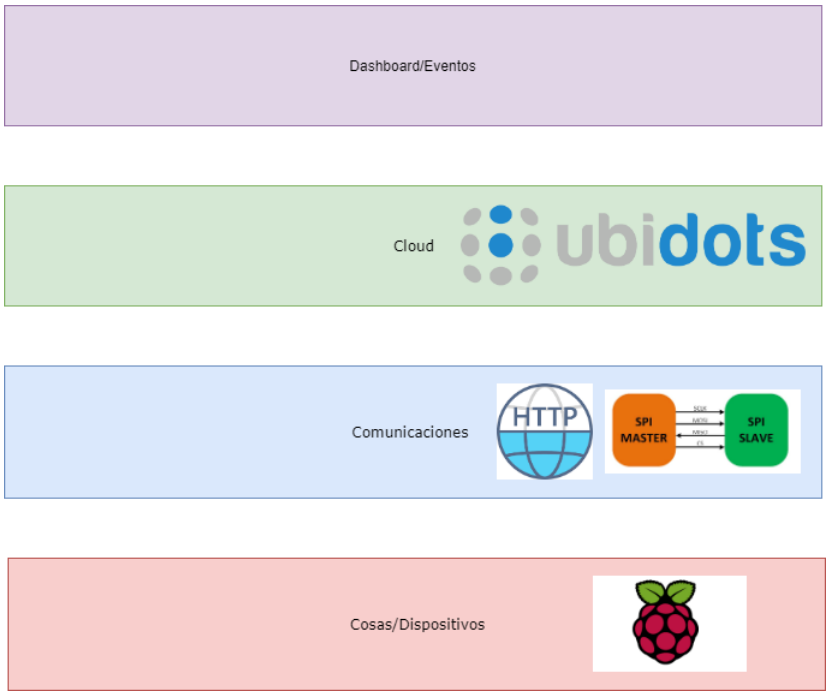


Figura 4.7: Diagrama explicativo de la arquitectura software del TFG.

5 Desarrollo.

En este apartado realizaremos una descripción detallada de cada uno de los pasos que se han seguido a lo largo del diseño del algoritmo central. Como se comentó anteriormente, todo el algoritmo está programado en Python.

5.1 Idea inicial del proyecto.

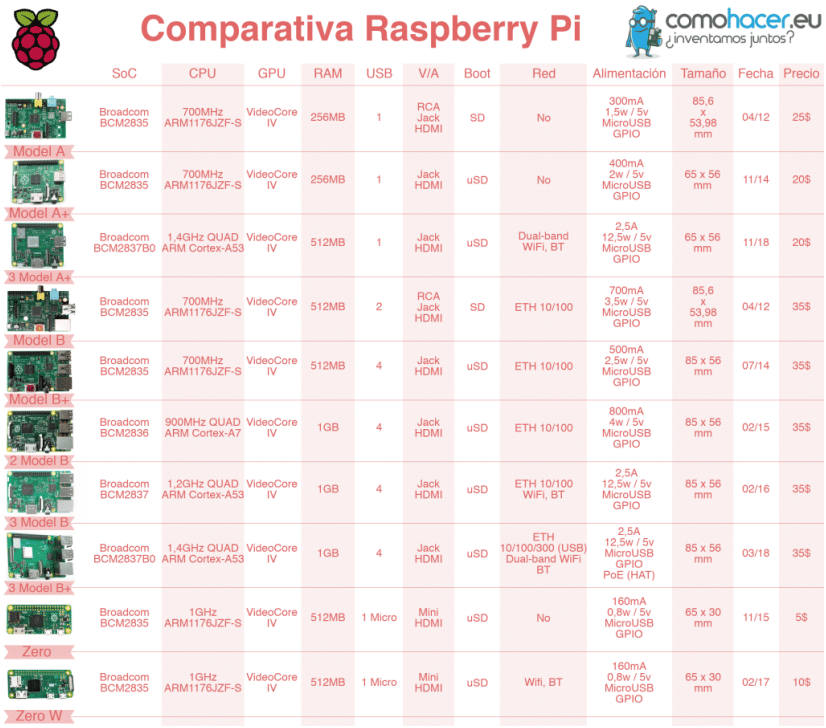
La idea surge a raíz de las prácticas en empresa en Elecgy Solutions, una empresa dedicada a la eficiencia energética, en la que estuve como becario durante 4 meses. La propuesta inicial era crear un gateway, aunque también se conocen como pasarela o puerta de enlace; cuya función sería la propia de estos dispositivos en el ámbito de la informática, compartir datos entre diversos dispositivos o servir de interfaz de conexión entre dispositivos. Un ejemplo cotidiano de gateway es el caso del router ADSL que tenemos todos en casa, el cual sirve de punto de conexión entre internet y nuestra red local doméstica.

Siendo más específicos y de cara a un ambiente industrial, existen muchísimos tipos de gateways en el mercado, como por ejemplo:

- Audiocodes.
- Cisco.
- Digium.
- Grandstream.
- Mediatrix.

Tras la idea de crear un gateway, surgió la idea de incorporar servicios o sistemas cloud, de manera que el sistema embebido que empleáramos sirviera de punto de conexión entre el sistema y la nube. Para añadir el apartado cloud al proyecto, se optó por el sistema Ubidots,

Figura 5.1: Gateway Mediatrix, de tipo analógico. Fuente: `https://www.usa-voip.com/shop/gateway-mediatrix-3621-1-pri-15-llamadas?___store=spanish&___from_store=english`



	SoC	CPU	GPU	RAM	USB	V/A	Boot	Red	Alimentación	Tamaño	Fecha	Precio
Model A	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	256MB	1	RCA Jack HDMI	SD	No	300mA 1,5w / 5v MicroUSB GPIO	85,6 x 53,98 mm	04/12	25$
Model A+	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	256MB	1	Jack HDMI	uSD	No	400mA 2w / 5v MicroUSB GPIO	65 x 56 mm	11/14	20$
3 Model A+	Broadcom BCM2837B0	1,4GHz QUAD ARM Cortex-A53	VideoCore IV	512MB	1	Jack HDMI	uSD	Dual-band WiFi, BT	2,5A 12,5w / 5v MicroUSB GPIO	65 x 56 mm	11/18	20$
Model B	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	512MB	2	RCA Jack HDMI	SD	ETH 10/100	700mA 3,5w / 5v MicroUSB GPIO	85,6 x 53,98 mm	04/12	35$
Model B+	Broadcom BCM2835	700MHz ARM1176JZF-S	VideoCore IV	512MB	4	Jack HDMI	uSD	ETH 10/100	500mA 2,5w / 5v MicroUSB GPIO	85 x 56 mm	07/14	35$
2 Model B	Broadcom BCM2836	900MHz QUAD ARM Cortex-A7	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100	800mA 4w / 5v MicroUSB GPIO	85 x 56 mm	02/15	35$
3 Model B	Broadcom BCM2837	1,2GHz QUAD ARM Cortex-A53	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100 WiFi, BT	2,5A 12,5w / 5v MicroUSB GPIO	85 x 56 mm	02/16	35$
3 Model B+	Broadcom BCM2837B0	1,4GHz QUAD ARM Cortex-A53	VideoCore IV	1GB	4	Jack HDMI	uSD	ETH 10/100/300 (USB) Dual-band WiFi BT	2,5A 12,5w / 5v MicroUSB GPIO PoE (HAT)	85 x 56 mm	03/18	35$
Zero	Broadcom BCM2835	1GHz ARM1176JZF-S	VideoCore IV	512MB	1 Micro	Mini HDMI	uSD	No	160mA 0,8w / 5v MicroUSB GPIO	65 x 30 mm	11/15	5$
Zero W	Broadcom BCM2835	1GHz ARM1176JZF-S	VideoCore IV	512MB	1 Micro	Mini HDMI	uSD	Wifi, BT	160mA 0,8w / 5v MicroUSB GPIO	65 x 30 mm	02/17	10$

Figura 5.2: Comparativa entre los distintos modelos que ofrece Raspberry. Fuente: `https://comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/`

ya que ya había trabajado previamente con él. Una vez añadido esto, ya fueron surgiendo ideas por el camino, como la capacidad de almacenar datos en un formato estándar en caso de no existir conexión a internet.

5.2 Selección del sistema embebido.

La única herramienta hardware que ha sido necesario seleccionar ha sido el sistema embebido, que en un principio se iba a emplear el sistema embebido Modberry 500, de la empresa TechBase [26]. Emplea un sistema operativo Raspbian basado en Debian Linux. Al principio del desarrollo, a la hora de leer datos de consumo, surgieron diversos problemas para acceder a los registros del ADC, ya que en el datasheet y en la página del sistema no aparecía qué ADC era. Este problema provocó un cierto retraso, y que al final acabó provocando que se emplease el sistema embebido Raspberry Pi 3, ya que contaba con más ayudas de este tipo y una gran comunidad detrás, de cara a búsqueda de códigos de ejemplo. Como se ha mencionado anteriormente, Raspberry es un sistema embebido TIC, mientras que el Modberry 500 sería de gama industrial, lo que, por otro lado; aumenta el precio del sistema, teniendo un precio 4 veces mayor que el de Raspberry. También cabe resaltar que dentro de Raspberry, existen diversos modelos, en mi caso elegí el modelo B+.

De todos modos, cabe resaltar que el sistema Modberry 500 es un buen sistema para este tipo de proyectos (gestión energética, redes inteligentes, casas inteligentes). Además, su versatilidad en el apartado de comunicaciones abre una gran cantidad de puertas, contando

Figura 5.3: Protocolos de comunicación de Modberry 500.

con protocolos como:

- Modbus.
- LORA.
- ZigBee.
- Bluetooth.
- WiFi.

5.3 Lectura de datos.

Para la lectura de los datos de consumo, era necesario emplear alguna librería que nos permitiese acceder al ADC y leer datos del mismo. Se ha empleado el bus Serial Peripheral Interface (SPI) [27]. Se emplea sobre todo para la comunicación entre circuitos integrados en equipos electrónicos, además de ser un interfaz de datos síncrono y que es un bus maestro-esclavo. Cuenta con 4 señales en total:

- SCL: el reloj, marca la sincronización.
- MOSI (Master Out/Slave In): es la salida de datos del maestro y la entrada de datos del esclavo.
- MISO (Master In/Slave Out): es la entrada de datos del maestro y la salida de datos del esclavo.
- SS/Select: es donde el maestro elige al esclavo.

Es un bus que nos brinda una buena velocidad de transmisión, comunicación dúplex[1]omunicación dúplex significa que se permite comunicación en ambos sentidos a la vez. y, sobre todo, simplicidad a la hora de integrarlo en algún algoritmo, a cambio de un mayor consumo que otros buses como, por ejemplo, I2C.

En las siguientes figuras se presentan varias representaciones de comunicación mediante SPI, tanto de un solo maestro a un solo esclavo como de un maestro a varios esclavos.

Figura 5.4: Caso de comunicación en SPI con un maestro y un solo esclavo. Fuente: `https://es.wikipedia.org/wiki/Serial_Peripheral_Interface`

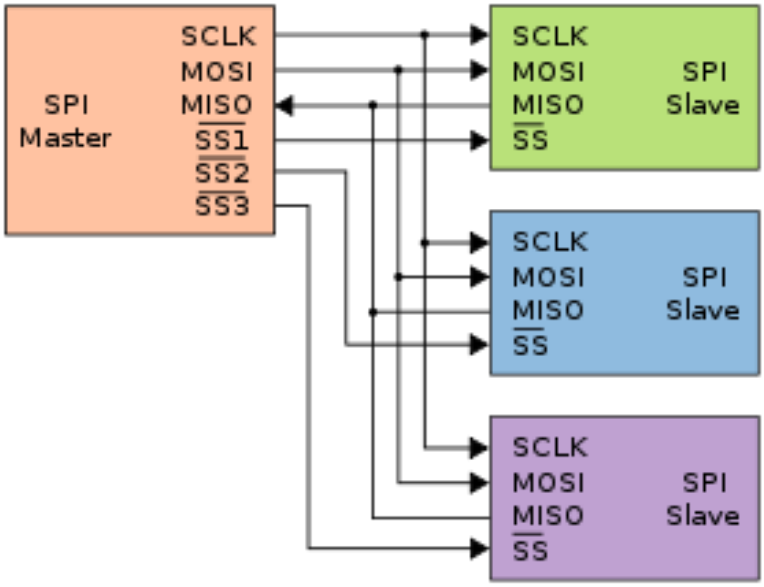


Figura 5.5: Caso de comunicación en SPI con un maestro y varios esclavos. Fuente: `https://es.wikipedia.org/wiki/Serial_Peripheral_Interface`

A la hora de implantar este protocolo en Python, existe una librería específica para SPI. La librería se llama spidev, y emplea el driver SPI que existe en el kernel de Linux.

Podemos observar un simple ejemplo de funcionamiento de la librería para abrir el registro de un dispositivo y escribir sobre él.

Código 5.1: Ejemplo básico de uso de SPIDev

```
import spidev
spi = spidev.SpiDev()
spi.open(bus, device)
to_send = [0x01, 0x02, 0x03]
spi.xfer(to_send)
}
```

Como observamos, la función open recibe el bus y el dispositivo que vamos a abrir para leer o escribir datos. En nuestro caso se integró un ADC en la el sistema embebido, el cual posee cuatro registros (del 0 al 3), y el bus es el número 0. En el código final, tan solo abrimos el registro como se ha observado en el código anterior, y empleamos dos funciones, llamadas ReadInput, que recibe el número de registro que queremos leer y devuelve el dato que hay almacenado en el mismo, y read data, que realiza la conversión del dato que leemos en crudo del ADC a voltaje. A continuación se muestran los códigos:

Código 5.2: Ejemplo básico de uso de SPIDev

```
def ReadInput(Sensor):
    adc = spi.xfer2([1,(8+Sensor)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2]
    return data

def read_data_1():
    for i in range(0,500):
    RawValue = ReadInput(0) #Read the raw input from the chip
    date = time.time()
    percent = (RawValue/10.24) #Convert to a percentage
    Voltage = (percent /100.0)*3.3 #Convert % to a 0 to 3.3 Volts
    vector_1[i] = Voltage
    vector_date_1[i] = date

def read_data_2():
    for i in range(0,500):
    RawValue = ReadInput(1) #Read the raw input from the chip
    date = time.time()
    percent = (RawValue/10.24) #Convert to a percentage
    Voltage = (percent /100.0)*3.3 #Convert % to a 0 to 3.3 Volts
    vector_2[i] = Voltage
    vector_date_2[i]=date

def read_data_3():
    for i in range(0,500):
    RawValue = ReadInput(2) #Read the raw input from the chip
    date = time.time()
    percent = (RawValue/10.24) #Convert to a percentage
    Voltage = (percent /100.0)*3.3 #Convert % to a 0 to 3.3 Volts
    vector_3[i] = Voltage
    vector_date_3[i] = date
```

Observamos que la función read data lleva un número asociado, esto es debido a que declaramos los vectores de manera global, y tenemos una función para leer cada una de las

[1] C

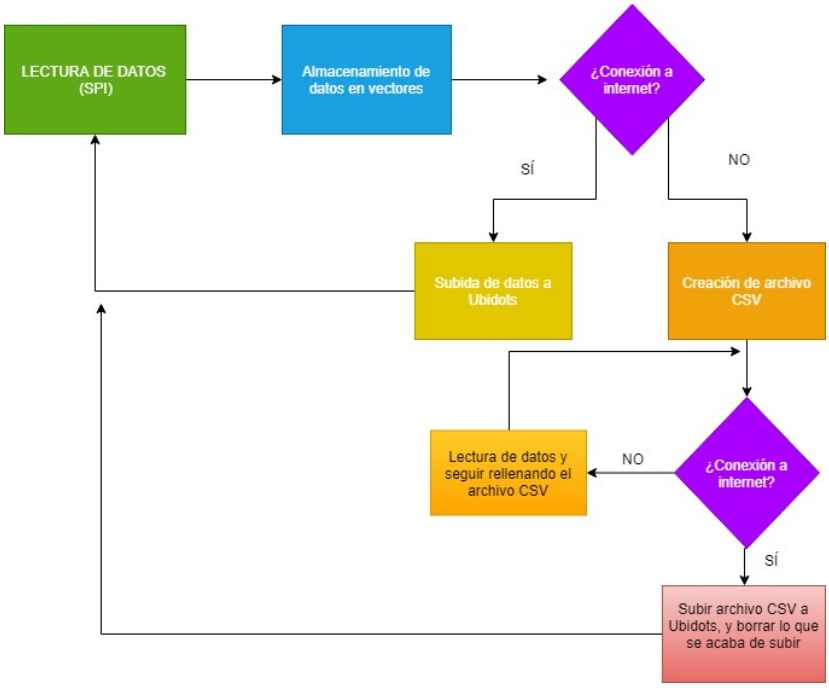


Figura 5.6: Grafo explicativo del algoritmo.

fases, ya que estamos leyendo de una red trifásica. La línea final sirve para almacenar la hora a la que se leyó el dato, empleando la librería datetime. La función time.time() devuelve el tiempo en segundos desde que el tiempo empieza, que suele ser el día 1 de enero del año, y este año varía en función del sistema, por ejemplo para Unix es 1970. Para saber desde donde comienza el tiempo, empleamos la función gmtime(0), también propia de la librería datetime. En caso de requerirse más información, desde la misma página de Python aparece información más detallada de la librería.

5.4 Grafo de funcionamiento del algoritmo.

El funcionamiento del algoritmo es simple, y podemos observarlo en la figura 5.6:

5.5 Subida de datos a la nube.

La parte cloud que incluye el trabajo de fin de grado consta de subir estos datos a la nube y emplear las herramientas que ésta nos brinda para tratar los datos, estas herramientas serán tratadas más adelante en las próximas secciones.

Como bien hemos mencionado la plataforma empleada ha sido Ubidots, y para poder comenzar a emplear nuestra propia nube, será necesario que nos creemos una cuenta dentro de la misma app. Una vez que obtengamos nuestra cuenta, tendremos que obtener el TOKEN, que viene a ser como la manera que tendremos de identificar nuestra nube del resto a la hora de realizar la subida de datos. Para obtenerlo, accedemos a nuestro perfil, y a la parte que

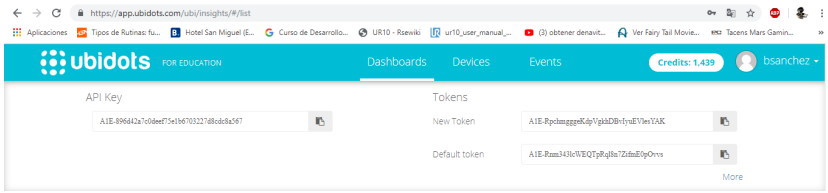


Figura 5.7: Nuestro TOKEN estaría definido por los caracteres que aparecen debajo de API Key, junto con la posiblidad de generar nuevos TOKEN.

pone API Credentials. En la siguiente figura se puede observar dónde aparece nuestro TOKEN identificativo.

Una vez que disponemos de nuestro TOKEN, en mi caso he optado por emplear la librería HTTP Requests para cubrir el apartado de comunicación con la nube. Una de las principales razones por las que he seleccionado este método es por la simplicidad que tiene a la hora de realizar las comunicaciones mediante HTTP, puesto que no se trata de un protocolo muy sencillo, y mediante esta librería se simplifica bastante el trabajo.

HTTP proviene de las siglas Hypertext Transfer Protocol, y es un protocolo que permite la transmisión de datos por el World Wide Web (WWW). Se trata de un protocolo que funciona empleando el esquema pregunta/respuesta entre un cliente y un servidor. En nuestro caso nosotros seríamos el cliente y Ubidots sería el servidor. Los mensajes en HTTP son texto plano, lo que mejora considerablemente la depuración, pero hace más largos los mensajes. Los mensajes están formados por [28]:

- Línea inicial, la cual varía en función de si es para petición o para respuestas:
  - Para peticiones: método de petición + URL + versión de HTTP.
  - Para respuestas: versión de HTTP + código de respuesta (indica qué ha pasado con la petición seguido de la URL del recurso) + frase asociada al retorno.
- Cabecera. Se trata de metadatos.
- Cuerpo del mensaje. Suelen ser los datos que se intercambian entre cliente y servidor.

HTTP cuenta con una gran cantidad de métodos de petición, que en nuestro caso los más empleados serán GET y POST, leer y subir datos a la nube, respectivamente. También cuenta con más métodos de petición, como PUT, DELETE, PATCH...

En nuestro código, disponemos de varias funciones para realizar la subida de datos, que son las siguientes:

Código 5.3: HTTP Requests

```
def post_request(payload):
    # Creates the headers for the HTTP requests
    url = "http://things.ubidots.com"
    url = "{}/api/v1.6/devices/{}".format(url, DEVICE_LABEL)
    headers = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}

    # Makes the HTTP requests
    status = 400
    attempts = 0
    while status >= 400 and attempts <= 5:
```

```
    req = requests.post(url=url, headers=headers, json=payload)
    status = req.status_code
    attempts += 1
    time.sleep(1)

    # Processes results
    if status >= 400:
    print("[ERROR] Could not send data after 5 attempts, please check \
    your token credentials and internet connection")
    return False

    print("[INFO]
    request made properly, your device is updated")
    return True


def build_payload(variable_1,variable_2,variable_3, valor_1, valor_2, valor_3):

    payload = {variable_1: valor_1,
    variable_2: valor_2,
    variable_3: valor_3}

    return payload
```

Observamos que la función build payload recibe una variable y un valor, esto es porque dentro de la nube cada variable tiene un nombre asignado, y asociamos el valor en esa función, para posteriormente pasarlo a post request que se encargará de subirlo a la nube.

5.6 Lectura de datos desde la nube.

Como se mencionó en la sección anterior, para la comunicación entre nube y sistema embebido empleamos la librería HTTP Requests. La lectura de datos desde la nube nos sirve para tener constancia de algún cambio mediante los eventos que nos permite crear Ubidots, los cuales comentaremos en la sección 5.8.

Estos eventos nos permiten alterar el valor de variables internas de la nube en función de eventos, como por ejemplo cambiar entre 1 y 0 para saber que ha sucedido un evento, como que por ejemplo algún consumo ha superado algún umbral que de cara a la empresa supondría algo ineficiente, mediante la lectura de esta variable sabríamos que si leemos un 1 es que esto ha pasado, y ya enviaríamos la información necesaria al usuario o tomaríamos las decisiones necesarias en el sistema embebido.

La lectura de datos desde la nube es muy simple, empleamos la función GET. El código para realizar esto sería el siguiente:

Código 5.4: Ejemplo función GET.

```
import requests
import time

TOKEN = "A1E-Rnm343lcWEQTpRql8n7ZifmE0pOvvs" #TOKEN for uploading data to ubidots
DEVICE_LABEL = "MEDIDOR_UA"
VARIABLE_LABEL_1 = "fan"

def main():

    fan = requests.get("http://things.ubidots.com/api/v1.6/devices/"+DEVICE_LABEL+"/"+↩
        ↪ VARIABLE_LABEL_1+"/lv?token="+TOKEN)

```

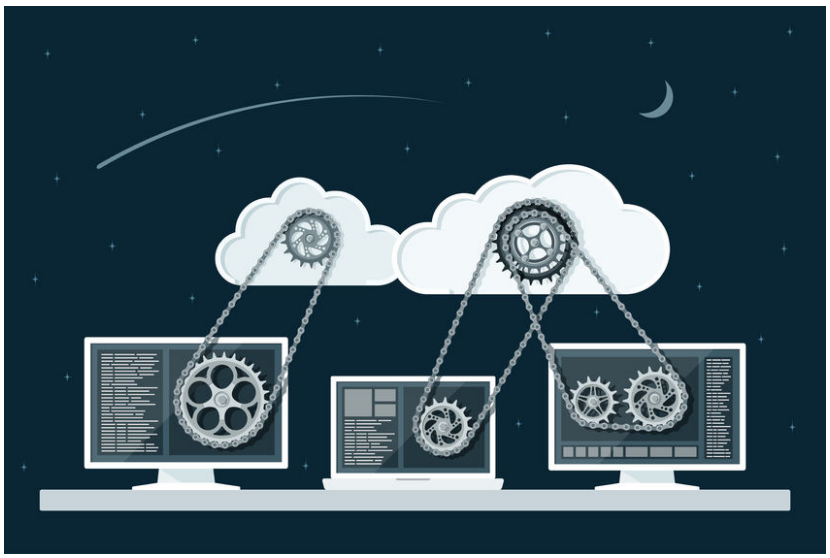


Figura 5.8: Lectura de datos en la nube. Fuente: `https://grupogaratu.com/dbaas-bases-datos-la-nube/`

```
12    if fan== "1.0":
13        print("Fan vale 1.")
14    elif fan == "0.0":
15        print("Fan vale 0.")
```

Observamos que tan solo realizamos una llamada a GET, junto con la etiqueta que tenga el dispositivo que queramos leer y su etiqueta, además del TOKEN del que ya hablamos anteriormente. Observamos que leemos la variable en formato string, por lo que para realizar algún tipo de operación matemática sería necesario aplicarle una conversión de tipo.

Emplear la nube de esta manera también podría ser de gran utilidad a la hora de comunicar dos sistemas, como por ejemplo que otro sistema embebido leyese la nube a la que sube datos otro sistema embebido, como se ilustra en la figura 5.7.

5.7 Almacenamiento de datos en local.

Existen ocasiones en las cuales en una empresa o laboratorio se producen caídas de conexión, lo que produce una gran pérdida de datos, ya que se imposibilita la subida de éstos a la nube y por lo tanto no se almacenan. Una idea que pensé añadir al sistema embebido fue la capacidad de almacenar estos datos en un formato estándar en caso de que no hubiese conexión a internet.

Primero comentaremos los formatos estándar para almacenamiento de datos más populares:

- Comma-Separated Values (CSV).
- GeoCSV. Se trata de una especificación reciente de CSV que almacena datos de geometría.
- JavaScript Object Notation (JSON).

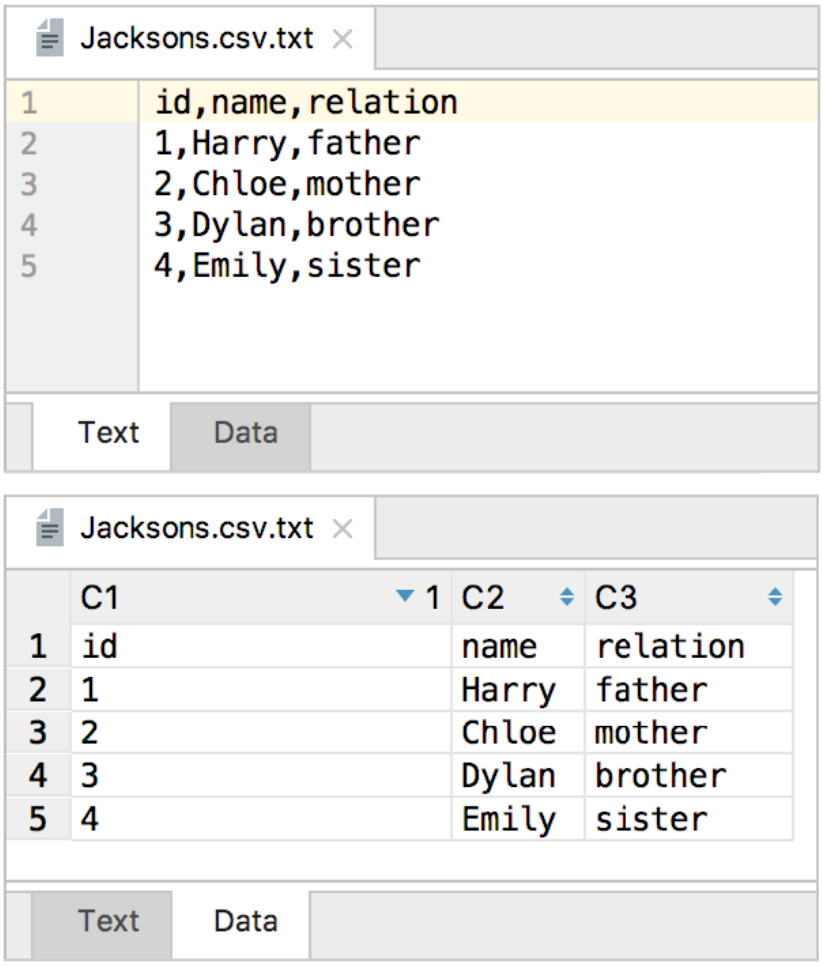


Figura 5.9: Ejemplo de CSV. Fuente: `https://www.jetbrains.com/help/clion/editing-csv-and-tsv-files.html`

CSV es un formato de documento de tipo abierto que representa los datos en forma de tabla separándolos por comas, y separando las filas por saltos de línea. Es muy simple pero no funciona muy bien en caso de que el dato que queramos guardar en la tabla contenga comillas o comas. Se presenta un simple ejemplo en la figura 5.8.

JSON es un formato de texto sencillo para intercambio de datos. Es una de las mejores alternativas a otros formatos como XML. Cuenta con muchos datos disponibles, entre ellos:

- Números. Se aceptan números negativos y fraccionados por puntos.
- Cadenas.
- Booleanos.
- NULL.
- Arrays.

- Objetos. Tienen la forma <nombre>:<valor>.

A continuación se presenta un ejemplo de almacenamiento de datos JSON:

Código 5.5: Ejemplo JSON.

```
{
   "menu": {
          "id": "file",
          "value": "File",
          "popup": {
                 "menuitem": [
                 {
                     "value": "New", "onclick": "CreateNewDoc()"
                 },{
                     "value": "Open", "onclick": "OpenDoc()"
                 },{
                     "value": "Close", "onclick": "CloseDoc()"
                 }
             ]
          }
      }
}
```

En mi caso he preferido emplear el formato CSV, por la simplicidad que aporta y la capacidad de exportarlo a otros programas como Microsoft Excel, así como la facilidad de lectura que tiene el formato, para poder resubir estos datos a la nube.

Una vez comentados los formatos de almacenamiento de datos, explicaré el código que se encarga de almacenar estos datos en un CSV y cómo se lee el archivo para resubir estos datos a la nube. En primer lugar se realiza la comprobación de la conexión a internet, para ello empleamos la librería de Python *URLLIB2*, la cual nos permite hacer un PING a una URL (en mi caso empleo una URL de Google), y en caso de que se reciba algo de vuelta, será que existe conexión a internet. Si existiera conexión, se seguiría el proceso normal de subida de datos a la nube, pero si no existe conexión, es cuando almacenamos los datos. Python cuenta con una librería de CSV que simplifica bastante el uso del mismo. La estructura que presenta el CSV es simple, en cada fila tenemos la lectura de cada trama y la fecha en la que se obtuvo el dato.

Para leer el CSV, empleamos la librería Pandas de Python, la cual es necesario instalar de manera externa. Es una librería destinada al análisis de datos, y que permite trabajar con ellos de manera eficiente. Para instalarla empleamos el administrador de paquetes (Pip), y sólo es necesario escribir la siguiente línea en la consola:

```
pip install pandas
```

Una vez que tenemos Pandas, leer el archivo CSV y trabajar con el se vuelve muy sencillo, ya que podemos extraer columnas y convertirlas en vector con un solo comando. Una vez que tenemos los datos en un vector, solo es necesario volver a realizar un POST y resubirlos a la nube. Una vez que se han resubido todos los datos, eliminamos el archivo, ya que cada vez que se recupera la conexión a internet se comprueba siempre si el archivo está vacío, lo que quiere decir que hay datos en local que aún no se han subido. Para todo este tratamiento de archivos, empleamos la librería *OS* de Python, que nos brinda funciones como comprobar si un archivo está vacío, borrarlo, u obtener directorios.

El código que realiza todo este proceso es:

Código 5.6: Código de almacenamiento de datos en local.

```
#generateCSV creates a CSV file for keeping local data in case of internet failures.
def generateCSV(vector_1,vector_2,vector_3):
    headerCsv = ['Row','Trama 1','Date','Trama 2','Date','Trama 3','Date']
    with open('data.csv','w') as csvFile:
    writer=csv.writer(csvFile)
    writer.writerow(headerCsv)
    for i in range(0,500):
    row = [i,vector_1[i],vector_date_1[i],vector_2[i],vector_date_2[i],vector_3[i],vector_date_3[i]]
    writer.writerow(row)
    csvFile.close()

#test internet connection making a PING to a Google IP address
def internet_on():
    try:
    urllib2.urlopen('http://216.58.192.142',timeout=1)
    return True
    except urllib2.URLError as err:
    return False

#test if CSV file is empty
def test_empty():
    if os.path.isfile("data.csv") == False:
    return True
    else:
    return False

#deletes CSV file
def delete_csv():
    remove("data.csv")
    print "CSV removed and updated on cloud."

#updates CSV file to cloud
def update_csv():
    data=pd.read_csv("data.csv")
    data_trama1 = data['Trama 1'].values
    data_trama2 = data['Trama 2'].values
    data_trama3 = data['Trama 3'].values

    for i in range(0,500):
    payload = build_payload(VARIABLE_LABEL_1,VARIABLE_LABEL_2,VARIABLE_LABEL_3,↩
        ↪ data_trama1[i],data_trama2[i],data_trama3[i])
    post_request(payload)
    print("[INFO] finished")

#main
def main():
    read_data_1()
    read_data_2()
    read_data_3()

    ret = internet_on()
    if ret == True:
        for i in range(0,500):
            payload = build_payload(VARIABLE_LABEL_1,VARIABLE_LABEL_2,↩
                ↪ VARIABLE_LABEL_3,vector_1[i],vector_2[i],vector_3[i])
            print("[INFO] Attemping to send data")
            post_request(payload)
            print("[INFO] finished")
            print "Checking if there is local data available for update."
            empty = test_empty()
            if empty == True:
                print "No local data available."
            else:
```

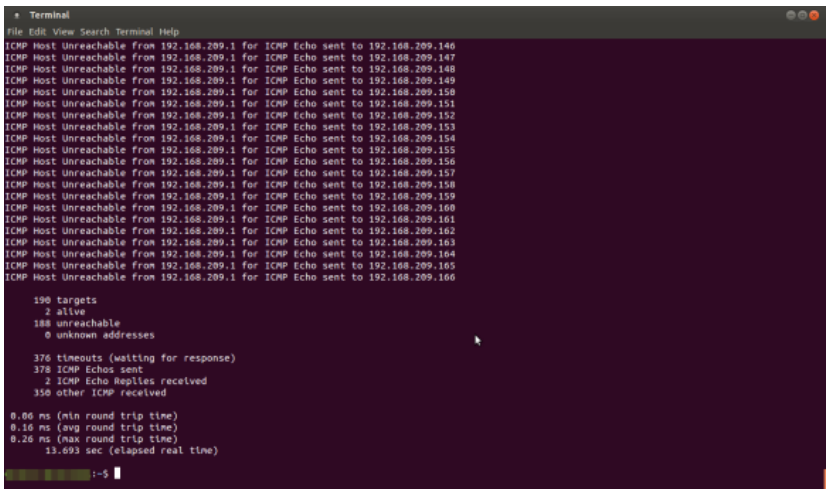


Figura 5.10: Ejemplo de uso de la herramienta PING en Linux.

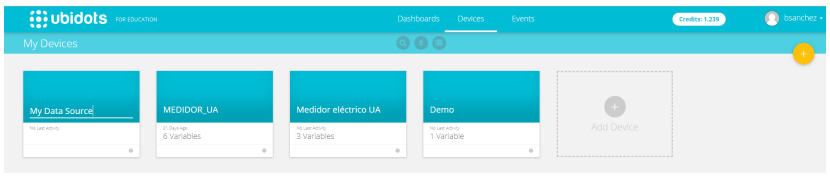


Figura 5.11: Observamos que al pulsar el botón + amarillo, añadimos un dispositivo nuevo, con el nombre predeterminado My Data Source.

```
62                print "Updating local data."
63                update_csv()
64                delete_csv()
65    else:
66            print("No internet connection, generating a CSV file with the last 500 values.")
67            generateCSV(vector_1,vector_2,vector_3)
68            print("CSV created.")
69
70
71
72if __name__ == '__main__':
73    while (True):
74    main()
75    time.sleep(1)
```

5.8 Herramientas en la nube.

Como se ha mencionado en capítulos anteriores, Ubidots nos brinda una gran cantidad de herramientas dentro de su aplicación IoT para tratar los datos. En primer lugar es necesario que añadamos un dispositivo, desde la pestaña Devices (figura 5.10), y añadamos variables a ese dispositivo, que será donde almacenaremos los datos que subamos.

Para añadir una herramienta en la aplicación tan solo tenemos que acceder a la aplicación y al apartado Dashboards, y pulsar sobre el símbolo + amarillo que observamos en la esquina superior derecha.

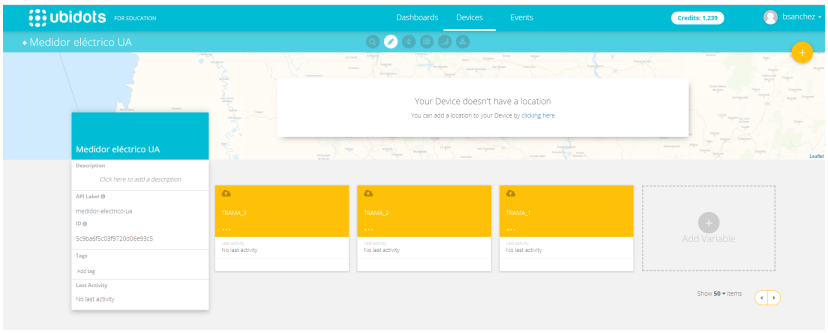


Figura 5.12: En esta ventana se añaden las variables, y la localización del dispositivo, que nos será muy útil de cara a herramientas como Map.

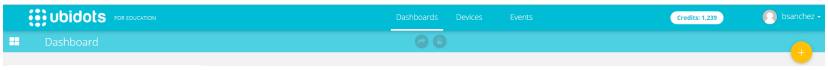


Figura 5.13: Una vez pulsemos sobre el símbolo +, se nos desplegarán todas las herramientas posibles.

Disponemos de varias herramientas:

- Chart. Nos permite realizar gráficas, tanto de líneas, barras e histogramas.
- Metric. Se trata de una de las herramientas de mayor utilidad, ya que nos permite calcular el sumatorio, media, máximo, mínimo, cuenta de datos, entre otros; de un conjunto de datos en un periodo de tiempo.
- Map. Esta herramienta requiere que le añadamos una ubicación a un dispositivo
- Table. Nos muestra los datos a modo de tabla, desde los últimos valores hasta un histórico de los mismos.
- Indicator. Nos permite visualizar los datos en forma de objetos, como On/Off, calibre o tanque.
- Control. Nos permite controlar variables desde la propia nube, pudiendo cambiar su estado entre On/Off o asignarle valores mediante un *slider*.
- HTML Canvas. Con esta herramienta podemos programar en HTML, de manera que se visualice tal y como nosotros deseemos. Ubidots nos ofrece varias plantillas desde su foro, como por ejemplo medidores en tiempo real.

En el siguiente código se muestra una plantilla de JS para hacer un medidor en tiempo real:

Código 5.7: Código de ejemplo para herramienta HTML Canvas.

```
1 var socket;
2 var srv = "industrial.ubidots.com:443";
3 var VAR_ID = "5a0f6e7a76254211cac86bea"; // Put here your var Id
4 var TOKEN = "..." // Put here your token
```

```
$( document ).ready(function() {
// Implements the connection to the server
socket = io.connect("https://"+ srv, {path: '/notifications'});
var subscribedVars = [];

// Function to publish the variable ID
var subscribeVariable = function (variable, callback) {
// Publishes the variable ID that wishes to listen
socket.emit('rt/variables/id/last_value', {
variable: variable
});
// Listens for changes
socket.on('rt/variables/' + variable + '/last_value', callback);
subscribedVars.push(variable);
};

// Function to unsubscribed for listening
var unSubscribeVariable = function (variable) {
socket.emit('unsub/rt/variables/id/last_value', {
variable: variable
});
var pst = subscribedVars.indexOf(variable);
if (pst !== -1){
subscribedVars.splice(pst, 1);
}
};

var connectSocket = function (){

// Implements the socket connection
socket.on('connect', function(){
socket.emit('authentication', {token: TOKEN});
});
window.addEventListener('online', function () {
socket.emit('authentication', {token: TOKEN});
});
socket.on('authenticated', function () {
subscribedVars.forEach(function (variable_id) {
socket.emit('rt/variables/id/last_value', { variable: variable_id });
});
});
}

/* Main Routine */

connectSocket();
// Should try to connect again if connection is lost
socket.on('reconnect', connectSocket());

// Subscribe Variable with your own code.
subscribeVariable(VAR_ID, function(value){
var parsedValue = JSON.parse(value);
console.log(parsedValue);
$('#content').text(value);
})
});
```

Y observamos un resultado como el de la figura 5.16.

Ubidots también nos permite la creación de eventos, para ello, clicamos en la pestaña Events y pulsamos en el + amarillo de la esquina superior derecha. Se trata de eventos simples, en los cuales tan solo podemos realizar comparaciones de las variables con un umbral. Entre las comparaciones que tenemos existen:

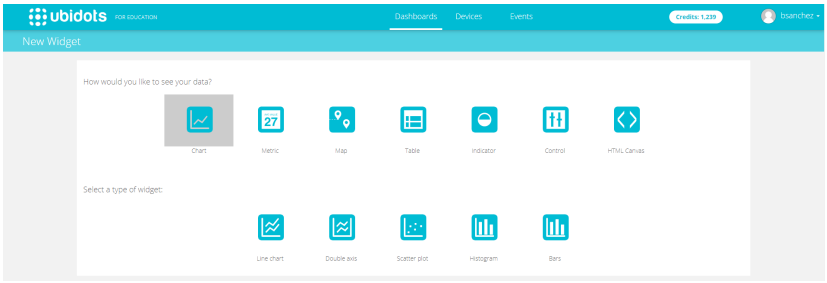


Figura 5.14: Posibilidades a la hora de graficar los datos.

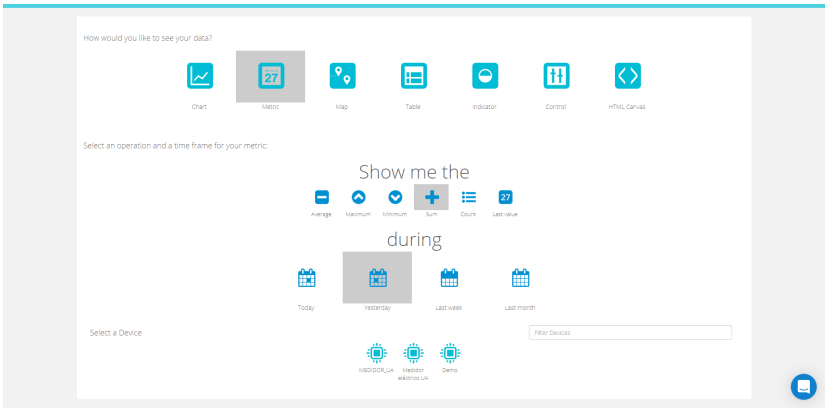
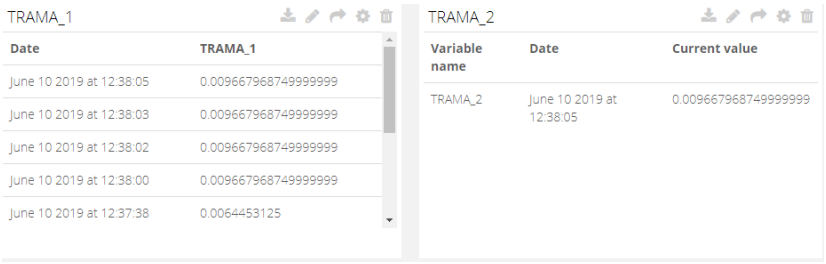


Figura 5.15: Podemos hacer métricas de los datos desde un solo día, hasta el mes anterior.



TRAMA_1

Date	TRAMA_1
June 10 2019 at 12:38:05	0.009667968749999999
June 10 2019 at 12:38:03	0.009667968749999999
June 10 2019 at 12:38:02	0.009667968749999999
June 10 2019 at 12:38:00	0.009667968749999999
June 10 2019 at 12:37:38	0.0064453125

TRAMA_2

Variable name	Date	Current value
TRAMA_2	June 10 2019 at 12:38:05	0.009667968749999999

Figura 5.16: A la izquierda observamos un histórico, y a la derecha tan sólo una tabla con los últimos valores.

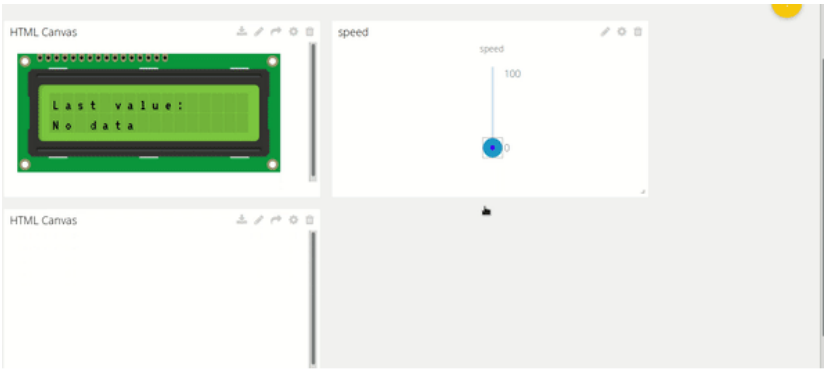


Figura 5.17: Resultado de la plantilla HTML Canvas.

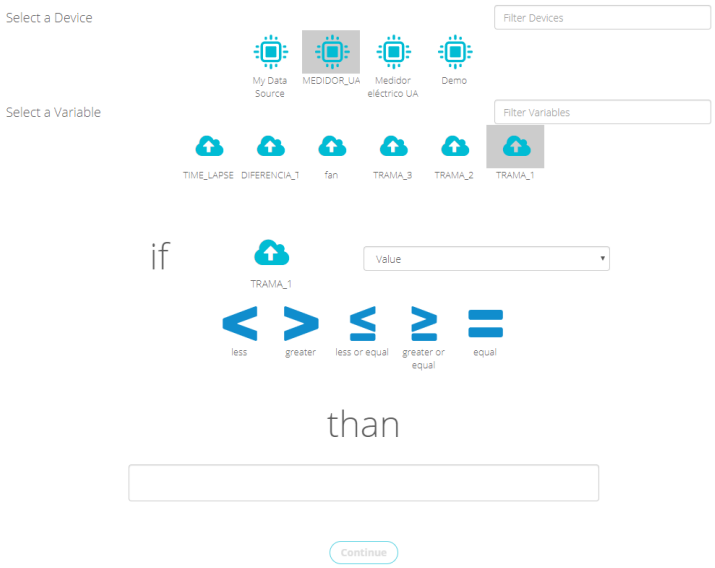


Figura 5.18: Pestaña de creación de eventos.

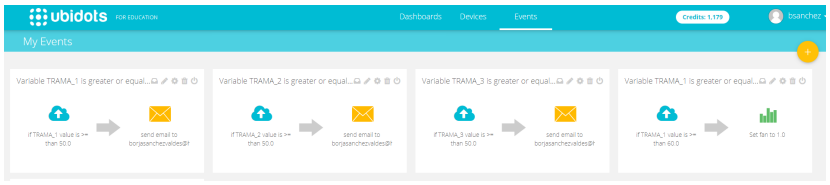


Figura 5.19: Ejemplos de varios eventos.

- Menor.
- Mayor.
- Mayor/Menor o igual.
- Igual.

Una vez que hemos creado la condición, podemos pedir que nos mande un sms, un correo electrónico, cambie el valor de una variable o envíe un mensaje en la aplicación Telegram [2].

En la figura 5.19 se presenta el modelo de correo electrónico que envía en caso de que suceda el evento.

5.9 Monitorización de generación de energías renovables.

Como añadido extra en los contenidos del TFG, se pensó incluir la monitorización de generación de energía renovable, para así poder realizar un mayor contraste entre lo que se genera y lo que se consume.

[2]Telegram es una aplicación de mensajería instantánea, similar a Whatsapp.

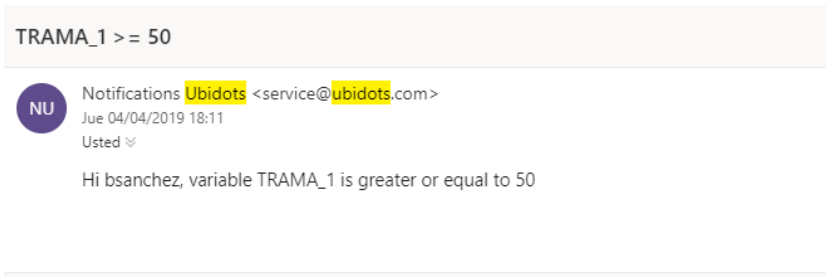


Figura 5.20: Ejemplo de correo electrónico en caso de que suceda un evento.

Figura 5.21: Observamos un valor nunca superior a 380-400 W, que es lo normal en generación de energía solar.

Las placas de energía solar se basan en el uso de inversores solares, que ya suelen comercializarse con su propio software de monitorización, aunque existen empresas que diseñan softwares de monitorización al gusto del cliente. En grandes rasgos, lo que hace un inversor solar es convertir corriente continua (DC) en corriente alterna (AC), de manera que podemos alimentar aparatos que funcionan con corriente alterna empleando corriente continua. Hoy en día estos inversores también cuentan con la capacidad de recopilar información sobre los niveles de producción y enviarla a los sistemas de monitorización fotovoltaica basados en la nube y a sus aplicaciones complementarias. [29]

En la figura 5.22 podemos observar el resultado de graficar las lecturas de generación de energía solar mediante un dashboard de Ubidots.

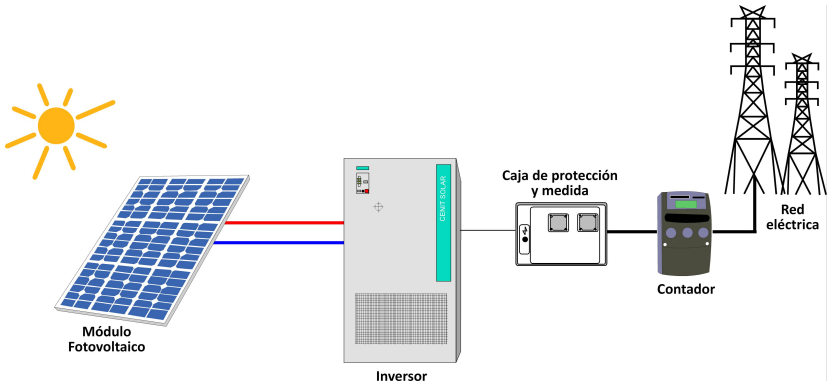


Figura 5.22: Imagen ilustrativa del uso de un inversor en instalaciones fotovoltaicas. Fuente: `http://www.cenitsolar.com/fotovoltaica_red_esquema.php`

6 Resultados.

En cuanto a los resultados obtenidos tras la finalización del TFG, haremos referencia a lo desarrollado y a lo que se ha obtenido finalmente, y a los beneficios que supone el desarrollo de un proyecto de esta índole.

6.1 Resultado final.

Una vez finalizado el proyecto, observamos que obtenemos un sistema formado por un sistema embebido TIC con un ADC que hemos añadido externamente, junto con una aplicación IoT que nos permite manejar los datos en la nube, entre las funcionalidades que posee destacamos:

- Leer datos de una red trifásica.
- Subir estos datos a la nube y monitorizar los valores de éstos.
- Capacidad de almacenar los datos en local, evitando así la pérdida de datos en caso de caídas de internet indeseadas.
- Gestionar los datos en la nube, empleando eventos que provocan el cambio de variables que están en constante lectura por parte del sistema embebido.

Además todo este sistema cuenta con un precio bastante económico.

6.2 Beneficios de la monitorización.

Monitorizar los valores de consumo nos es muy útil de cara a muchos factores [30, 31]:

- Desde un punto de vista empresarial, se necesita saber cuánto consumimos y llevar un historial de cuándo consumimos más y cuándo menos.
- Trabajar con todos estos valores de consumo y monitorizarlos puede servirnos de cara a desarrollar algún sistema autónomo capaz de tomar decisiones en función de cuánto se esté consumiendo.
- Tener consciencia del impacto ambiental que tiene nuestro consumo.
- Al saber cuánto consumimos, podemos planificar mejor las cargas de trabajo, de cara que no se supere la potencia.
- Esta monitorización puede funcionar como alarma en caso de fallo en alguna máquina.

Figura 6.1: Ventajas de la monitorización según Federico Arroyo.

- Detección de ineficiencias.

En la figura 6.1 se presentan más ventajas de la monitorización, se trata de una imagen obtenida del blog Proyectos y Certificaciones, perteneciente a Federico Arroyo.

6.3 Beneficios de la gestión local.

Gestionar los datos en local, tal y como hemos comentado, presenta la principal ventaja de evitar la pérdida de datos en caso de que no haya conexión a internet. Podríamos decir que es como un seguro de que si no tenemos conexión, los datos van a permanecer almacenados dentro del sistema embebido. Además, al almacenarlos en formatos como CSV, que tiene una gran facilidad de exportación, se nos abre un abanico de posibilidades de cara a trabajar con esos datos en otros programas además de en el apartado cloud.

6.4 Beneficios de la gestión en la nube.

Gestionar los datos en la nube tiene muchas ventajas, entre las que destacamos [32, 33]:

- Flexibilidad y movilidad, ya que desde cualquier dispositivo con internet se puede acceder a ella.
- Ahorro de espacio.
- Seguridad.
- Facilidad de transmisión de los datos.
- Efectividad.
- Ahorro energético, ya que no es necesario emplear máquinas ni aparatos electrónicos.
- Facilidad de implementación y simplicidad.

7 Conclusiones.

7.1 Estado actual del proyecto.

Podemos considerar que el sistema funciona a un 95 por ciento en referencia a los objetivos previos que se mencionaron, además de cumplir el objetivo general del TFG, que era conseguir un sistema que leyese los datos desde una red trifásica y los manejara en la nube, explotando todas las herramientas que nos da, en mi caso, la plataforma IoT Ubidots.

Cabe resaltar que todo el proyecto se ha realizado teniendo en cuenta posibles ideas de mejora o ampliaciones que se podrían añadir, teniendo en cuenta las opiniones del tutor académico y de la empresa Elecgy Solutions.

Se ha podido demostrar también que con un sistema embebido TIC se pueden desarrollar grandes aplicaciones y sin requerir una gran inversión económica.

7.2 Mejoras posibles para el proyecto.

Entre las posibles mejoras que se podrían añadir resaltamos:

- Capacidad de interactuar con actuadores u otros sistemas embebidos, añadiéndole la capacidad de tomar decisiones en caso de un alto consumo y dándole más autonomía.
- Empleabilidad de otros medios para comunicar con la nube, como, por ejemplo; MQTT.
- Añadir otro tipo de buses para comunicarnos con el ADC, como podría ser Inter-Integrated Circuit (I2C).
- Manejar otros tipos de datos, además de voltaje.
- Teniendo en cuenta el precio de la potencia contratada, hacer una aproximación del precio que se pagará en determinado mes por el consumo.
- Empleabilidad de otro tipo de formatos para almacenar los datos además de CSV, como por ejemplo JSON; aunque para el caso de datos de voltaje CSV era de los mejores por su facilidad de exportación a otros programas.

7.3 Nociones aprendidas.

Como se habrá podido apreciar, este TFG quizá no vaya muy relacionado estrechamente a la robótica como tal, ya que está más relacionado con comunicaciones, Big Data e IoT. El hecho de haber realizado un TFG más relacionado con este tipo de aspectos teniendo como base asignaturas como Comunicaciones o Sensores e Instrumentación ha hecho que se amplíen muchísimo más todos mis conocimientos en este tipo de herramientas que tanta importancia

están cogiendo, y que en combinación con robótica o inteligencia artificial tiene un amplio camino de desarrollo que podría dar lugar a aplicaciones de gran valor.

Un punto muy fuerte ha sido el aprendizaje de tecnologías o librerías que desconocía, como por ejemplo Latex, Pandas, o la librería *OS* de Python.

Haciendo referencia a problemas tecnológicos, no se han presentado muchos más allá del uso del sistema embebido Modberry, que, como se mencionó en capítulos anteriores, al ser de tipo industrial, quizá presentaba más dificultades de uso de cara a un estudiante como yo, que en plataformas como Raspberry prácticamente desaparecían.

7.4 Resultado y conclusión final.

Tras finalizar este TFG podemos decir que el resultado es plenamente satisfactorio, ya que se han conseguido todos los objetivos de manera correcta y tenemos un prototipo funcional.

Como siempre se menciona en este tipo de trabajos, me gustaría haber dedicado más tiempo para poder implementar muchas más funcionalidades, pero por falta del mismo no se han podido añadir. A pesar de esto, estoy contento con lo que se ha conseguido obtener y cómo ha quedado el prototipo. También cabe resaltar que es mi primer proyecto realizado prácticamente de forma individual, lo que complica un poco las cosas debido a la costumbre de realizar los trabajos en grupo.

Para finalizar, resaltar que a día de hoy la tecnología avanza a pasos agigantados y se puede apreciar en fenómenos como la Industria 4.0, y espero que este pequeño prototipo pueda despertar ideas a otros alumnos de cada a ampliarlo o aportar mejoras.

Bibliografía

[1] Grupo Proyectos y certificaciones. Beneficios del internet de las cosas en la eficiencia energética. http://proyectosycertificaciones.com/2016/08/internet-de-las-cosas-y-eficiencia/, 2016. Visitado: Enero 2019.

[2] Grupo Novelec. ¿cómo funciona una smart grid? https://blog.gruponovelec.com/electricidad/como-funciona-smart-grid/, 2017. Visitado: Enero 2019.

[3] Wikipedia. Red eléctrica inteligente. https://es.wikipedia.org/wiki/Red_eléctrica_inteligente#Antecedentes, 2019. Visitado: Febrero 2019.

[4] Comisión Europea. Paquete de medidas sobre clima y energía hasta 2020. https://ec.europa.eu/clima/policies/strategies/2020_es. Visitado: Febrero 2019.

[5] EcoInteligencia. Las smart grid y sus fundamentos. https://www.ecointeligencia.com/2014/02/smart-grid-fundamentos/, 2014. Visitado: Febrero 2019.

[6] Esteban Mauricio Inga Ortega. Redes de comunicación en smart grid. https://core.ac.uk/download/pdf/84704803.pdf, 2012. Visitado: Febrero 2019.

[7] Domótica Integrada. ¿qué son los edificios inteligentes y qué características los diferencian? https://domoticaintegrada.com/edificios-inteligentes/. Visitado: Febrero 2019.

[8] LonMark España. ¿qué es lon? http://www.lonmark.es/www/sistemas/queeslon.php?mn=21. Visitado: Marzo 2019.

[9] Controlli Delta Spain. Protocolo bacnet. http://www.controlli.es/protocolo-bacnet-automatizacion-de-edificios.html, 2017. Visitado: Marzo 2019.

[10] Samsung España. Smart things. https://www.samsung.com/es/apps/smartthings/. Visitado: Abril 2019.

[11] Construcción y rehabilitación. The edge, el edificio más sostenible e inteligente del mundo. http://construccionyrehabilitacion.com/2017/05/19/the-edge-edificio-mas-sostenible-e-inteligente-del-mundo/, 2017. Visitado: Marzo 2019.

[12] Plataforma arquitectura. The edge / plp architecture. https://www.plataformaarquitectura.cl/cl/790319/the-edge-plp-architecture, 2016. Visitado: Marzo 2019.

[13] Ciudades del futuro. The crystal: edificio icono de la sostenibilidad. https://ciudadesdelfuturo.es/the-crystal-edificio-icono-de-sostenibilidad.php, 2012. Visitado: Marzo 2019.

[14] Siemens. Desigo – la plataforma de gestión de edificios digital e integrada. https://new.siemens.com/es/es/productos/building-technologies/automatizacion/desigo.html. Visitado: Marzo 2019.

[15] Belén Rodrigo. La nueva imagen de torre europa, el rascacielos «más inteligente» de españa. Visitado: Marzo 2019.

[16] Wikipedia. Power over ethernet. https://es.wikipedia.org/wiki/Power_over_Ethernet, 2019. Visitado: Marzo 2019.

[17] Siemens. Simatic iot2040 | la pasarela inteligente para las soluciones de iot industrial. https://w5.siemens.com/spain/web/es/industry/automatizacion/simatic/pc_industriales/pages/simatic-iot2040.aspx. Visitado: Abril 2019.

[18] Wikipedia. Tecnologías de la información y la comunicaciónl. https://es.wikipedia.org/wiki/Tecnologías_de_la_información_y_la_comunicación. Visitado: Abril 2019.

[19] Components101. Raspberry pi 3. https://components101.com/microcontrollers/raspberry-pi-3-pinout-features-datasheet. Visitado: Abril 2019.

[20] Karel Gomez. Top 5 metodologías de desarrollo de software. https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/metodologias-de-desarrollo-de-software, 2017. Visitado: Marzo 2019.

[21] Wikipedia. Metodología de desarrollo de software. https://es.wikipedia.org/wiki/Metodología_de_desarrollo_de_software, 2019. Visitado: Marzo 2019.

[22] Wikipedia. Arquitectura de software. https://es.wikipedia.org/wiki/Arquitectura_de_software, 2019. Visitado: Abril 2019.

[23] Rubén Sánchez Corcuera. Iot ya está aquí. https://blogs.deusto.es/master-informatica/iot-ya-esta-aqui/, 2017. Visitado: Abril 2019.

[24] Wikipedia. Geany. https://es.wikipedia.org/wiki/Geany, 2018. Visitado: Abril 2019.

[25] Wikipedia. Texstudio. https://es.wikipedia.org/wiki/TeXstudio, 2019. Visitado: Abril 2019.

[26] TechBase. Modberry. https://modberry.techbase.eu/. Visitado: Abril 2019.

[27] Wikipedia. Serial peripheral interface. https://es.wikipedia.org/wiki/Serial_Peripheral_Interface, 2019. Visitado: Mayo 2019.

[28] Wikipedia. Protocolo de transferencia de hipertexto. https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto, 2019. Visitado: Mayo 2019.

[29] SotySolar. Monitorización de instalaciones de autoconsumo. https://sotysolar.es/placas-solares/monitorizacion. Visitado: Junio 2019.

[30] Creara Energy Experts. Ventajas de los sistemas de monitorización de consumo energético. https://www.creara.es/post/ventajas-sistemas-monitorizacion-energia, 2017. Visitado: Junio 2019.

[31] SinCeO2 Consultoría Energética. Monitorización de consumos energéticos. https://www.sinceo2.com/servicios/gestion-energetica-energia-electrica/monitorizacion-sistemas/. Visitado: Junio 2019.

[32] Universia Argentina. ¿cuáles son las ventajas de trabajar con la nube? https://noticias.universia.com.ar/ciencia-nn-tt/noticia/2014/08/14/1109707/cuales-ventajas-trabajar-nube.html#, 2014. Visitado: Junio 2019.

[33] Kyocera Document Solutions. Ventajas de trabajar en la nube. https://smarterworkspaces.kyocera.es/blog/ventajas-trabajar-en-la-nube/, 2017. Visitado: Junio 2019.